Transform safety-critical Cyber Physical Systems into distributed solutions for end-users and partners

D8 (D3.1)

Selection of concepts for end-to-end safety and performance for distributed CPS solutions

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007260. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Netherlands, Finland, Germany, Poland, Austria, Spain, Belgium, Denmark, Norway.

ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership

Document Information

Project	TRANSACT
Grant Agreement No.	101007260
Work Package No.	WP3
Task No.	T3.1
Deliverable No.	D8
Deliverable No. in WP	D3.1
Deliverable Title	Selection of concepts for end-to-end safety and performance for distributed CPS solutions
Nature	Report
Dissemination Level	Public
Document Version	v1.0
Date	30/05/2022
Contact	Teun Hendriks
Organization	TNO
Phone	+31 615484883
E-Mail	Teun.Hendriks@tno.nl

Authors Table

NAME	COMPANY	E-MAIL
Teun Hendriks (editor)	TNO	Teun.Hendriks@tno.nl
Behnam Asadi Khashooei	TNO	behnam.asadikhashooei@tno.nl
Benny Akesson	TNO	benny.akesson@tno.nl
Bjørn Åge Hjøllo	NVT	bjorn.hjollo@navtor.com
Carlo A. Boano	TUG	cboano@tugraz.at
Dusica Marijan	SRL	dusica@simula.no
Egbert Jaspers	VIN	egbert.jaspers@vinotion.nl
Elisabeth Salomon	TUG	elisabeth.salomon@tugraz.at
Gaurav Choudhary	DTU	gauch@dtu.dk
Ignacio Martinez	SNG	Ignacio.martinez@singlarinnovacion.com
Jarno Kallio	NOD	jarno.kallio@nodeon.com
Javier Coronel Parada	ITI	jcoronel@iti.es
Jeroen Voeten	TUE	J.P.M.Voeten@tue.nl
Joan Josep Valls Mompó	ITI	jvalls@iti.es
Jussi Litja	FLEET	jussi.litja@fleetonomy.ai
Jyrki Järvinen	NOD	jyrki.jarvinen@nodeon.com
Kilian Michiels	FEops	kilian.michiels@feops.com
Koen de Laat	PMS	koen.de.laat@philips.com
Krzysztof Oborzynski	PMS	krzysztof.oborzynski@philips.com
Marc Geilen	TUE	m.c.w.geilen@tue.nl
María José Hernández	SNG	mariajose.hernandez@nunsys.com
Miguel García Gordillo	ITI	miguelgarcia@iti.es
Mika Jaakonaho	FLEET	mika.jaakonaho@fleetonomy.ai
Mitra Nasri Nasrabadi	TUE	m.nasri@tue.nl
Paul Pop	DTU	paupo@dtu.dk
Peter Mortier	FEops	peter.mortier@feops.com
Sara Pastor	SNG	sara.pastor@nunsys.com
Sebastian Vander Maelen	DLR-SE	sebastian.vandermaelen@dlr.de

NAME	COMPANY	E-MAIL
Sergio Sáez	ITI	ssaez@iti.es
Steffen Renisch	PFLH	steffen.renisch@philips.com
Tetyana Kholodna	NVT	tetyana.kholodna@navtor.com

Reviewers Table

VERSION	DATE	REVIEWER
v0.8	19/04/2022	Mitra Nasri (Technische Universiteit Eindhoven)
v0.8	11/04/2022	Kenneth Sharman (Instituto Tecnologico de Informatica)
v0.8	19/04/2022	Marco Jahn (Eclipse Foundation Europe GmbH)
v0.9	20/05/2022	Sasa Marinkovic (Philips)

Change History

VERSION	DATE	REASON FOR CHANGE	AFFECTED PAGES
v0.1	28/07/2021	Initial structure of Deliverable D8 (D3.1)	All
v0.3	Various dates	Working document in progress	All
v0.8	18/03/2022	Consolidated version for TRANSACT review	All
v0.9	18/05/2022	Version after review for finalisation	All
v1.0	30/05/2022	Final version for submission	n/a

Table of Contents

List of Figures

List of Tables

1 Glossary

Table 1: Terms Definitions

TERM	DEFINITION
Allocation	Task allocation refers to the runtime decision of task placement and scheduling associated with the resource management.
Application	The functionality that implements a particular solution to help the end-user to perform a specific task. An application can be composed of a monolithic service or a group of distributed services which are executed in different and distributed targets in the device, edge, cloud continuum.
Architecture	The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.
Architecture framework	Conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders
Component	One of the parts that make up a system
Computing platform	A computing platform is the environment in which a piece of software is executed. It may be the hardware or the operating system (OS), even a web browser and associated application programming interfaces, or other underlying software, as long as the program code is executed with it. Computing platforms have different abstraction levels, including a computer architecture, an OS, or runtime libraries. A computing platform is the stage on which computer programs can run.
Concept	An abstraction; a general idea inferred or derived from specific instances
Cross-cutting concepts	System-level methods and techniques for linking application and platform. They include concepts for designing and deploying the application on the platform, as well as analysing and run-time monitoring and managing the behaviour of the application running on a specific platform in the device, edge, cloud continuum.
Cyber-Physical System	Digital system that semi-automatically interacts with its physical environment as integral part of its functionality.
Device	Physical entity embedded inside, or attached to, another physical entity in its vicinity, with capabilities to convey digital information from or to that physical entity
Method	A method consists of techniques for performing a task, in other words, it defines the “how” of each task.
Methodology	A collection of related processes, methods, and tools. A methodology is essentially a “recipe” and can be thought of as the application of related processes, methods, and tools to a class of problems that all have something in common.
Middleware	Middleware is computer software that provides services to software applications beyond those available from the operating system. It can be described as "software glue". Middleware makes it easier for software developers to implement communication and input/output, so they can focus on the specific purpose of their application.
Mission-critical system	A mission critical system is a system that is essential to the survival of a business or organization. When a mission critical system fails or is interrupted, business operations are significantly impacted.

TERM	DEFINITION
Orchestration	Type of composition where one particular element is used by the composition to oversee and direct the other elements. Note: the element that directs an orchestration is not part of the orchestration.
Partitioning	Divides the application code into several parts that will be executed on different platforms, i.e., mobile devices, cloudlets, or the cloud.
Platform	The environment in which the application is executed. It comprises of the complete infrastructure in the device, edge, cloud continuum to execute the application, including hardware, operating systems, hypervisors, communication networks, containers and cloud computing services.
Process	A process is a logical sequence of tasks performed to achieve a particular objective. A process defines “what” is to be done, without specifying “how” each task is performed.
Reference Architecture	A Reference Architecture (RA) is an architectural design pattern that indicates how an abstract set of mechanisms and relationships realizes a predetermined set of requirements. It captures the essence of the architecture of a collection of systems. The main purpose of a Reference Architecture is to provide guidance for the development of architectures. One or more reference architectures may be derived from a common reference model, to address different purposes/usages to which the Reference Model may be targeted.
Reference Model	A reference model is an abstract framework for understanding significant relationships among the entities of some environment. It enables the development of specific reference or concrete architectures using consistent standards or specifications supporting that environment. A reference model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or other concrete details. A reference model may be used as a basis for education and explaining standards to non- specialists.
Safety-critical system	A system whose failure or malfunction may result in one (or more) of the following outcomes: death or serious injury to people, loss or severe damage to equipment/property, environmental harm.
Service	Services are the mechanism by which needs and capabilities are brought together.
Solution	A means of solving a problem or dealing with a difficult situation.
System	A combination of interacting elements organized to achieve one or more stated purposes
System Component	A system architectural element.
Technique	Technical and managerial procedure that aids in the evaluation and improvement of the [system] development process.
Tool	A tool is an instrument that, when applied to a particular method, can enhance the efficiency of the task; provided it is applied properly and by somebody with proper skills and training.

Table 2: Abbreviations

ACRONYM	DEFINITION
ADS	Automated Driving System
AI	Artificial Intelligence
AIS	Automatic Identification System

ACRONYM	DEFINITION
API	Application Programming Interface
AV	Autonomous Vehicle
AWS	Amazon Web Services
BDaaS	Big Data as a Service
BMS	Battery Management System
CAN	Control Area Network
CCA	Clear Channel Assessment
DDS	Data Distribution Service
ECDIS	Electronic Chart Display and Information System
EVF	Electric Vehicle Fleet
FMA	Fleet Management Application (Fleetonomy.ai's proprietary remote operation platform)
FMEA	Failure Mode and Effects Analysis
FTA	Fault Tree Analysis
HAZOP	HAZard And OPerability study
HITL	Human-In-The-Loop
IoT	Internet of Things
ISO	International Organization for Standardization
ICT	Information and Communication Technology
ICS	Industrial Control Systems
IT	Information Technology
I2V	Infrastructure to Vehicle
KPI	Key Performance Indicator
LAN	Local Areas Network
LTE	Long Term Evolution
MAC	Medium Access Control
MD-SysPE	Model Driven System Performance Engineering
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
MR	Magnetic Resonance
ODD	Operational Design Domain
OEM	Original Equipment Manufacturer
PC	Personal Computer
PKI	Public Key Infrastructure

ACRONYM	DEFINITION
PLM	Product Lifecycle Management
PRR	Packet Reception Rate
QoS	Quality of Service
R&D	Research and Development
RF	Radio Frequency
SatCom	Satellite Communication
SC-CPS	Safety-Critical CPS
SIL	Safety-Integrity Level
SLA	Service Level Agreement
SLO	Service Level Objective
SoS	System of Systems
SoTA	State of The Art
SW	Software
TARA	Threat Analysis and Risk Assessment
TLEX	Traffic Live Exchange Platform
ToC	Transfer of Control
TRL	Technology Readiness Level
UC	Use Case
V&V	Verification & Validation
VIT	Virtual Integration Test
VM	Virtual Machine
V2V	Vehicle to Vehicle
WP	Work Package
WSN	Wireless Sensor Network
WWTP	Waste Water Treatment Plant

2 Introduction

This report is a result of the TRANSACT project, in specific a result of TRANSACT task 3.1. This task was to investigate use cases and technical requirements (originating from work package 1) and identify necessary concepts for managing performance and safety for deploying and running distributed applications for safety-critical Cyber-Physical Systems (CPS).

Specific attention points in this task's investigation were the following:

- Performance and safety monitoring requirements and concepts applicable to the TRANSACT reference architecture and distributed safety-critical applications;
- Requirements and concepts to predict (e.g. based on simulation) end-to-end safety and performance applicable to the TRANSACT reference architecture, including safety assurance concepts suitable for re-qualification (building on the concepts developed in past ECSEL projects to facilitate the independent development of safety functions);
- Concepts to manage run-time operation, including fail-safe concepts such as operational safeguards and degraded operation modes which ensure that there is always in-device functionality available in case the edge or cloud parts of the application is not responsive anymore;
- Risk analysis to assure the design of safety and performance controls for each component (in all scenarios and use cases) in the TRANSACT reference architecture to assure the same level of risk (based on simulation analysis);
- The impact and relation of safety and performance concepts on security and privacy and vice versa.

This deliverable reports on the results of these activities.

2.1 Role of the deliverable

This document has the following major purposes:

- Documentation of selected concepts for safety and performance analysis, to be applicable across the TRANSACT domains.
- Documentation of selected concepts to monitor and manage safety and performance in run-time operation, including fail-safe concepts
- Documentation of selected safety assurance concepts and risk assessment methods to validate a proper design of safety and performance controls, in support of (re-) qualification.

The results in this deliverable are closely aligned and harmonised with the 'sister' deliverable D3.2 targeting security and privacy concepts for distributed CPS solutions, and ensured to be fitting with the TRANSACT reference architectures as described in Deliverable D2.1.

2.2 Relationship to other TRANSACT documents

This document relates to the following TRANSACT deliverables:

- D5 (D1.1) Use case descriptions, end user requirements, SotA and KPI's (M10)

  The selected concepts for end-to-end safety and performance for distributed safety-critical CPS are aligned with the needs of the use cases as documented in D1.1 and will be applied in the context of those use cases. This deliverable includes a brief overview of the TRANSACT use cases.

- D6 (D1.2) Technical requirements and TRANSACT transition methodology commonalities (M12)

  The selected concepts for end-to-end safety and performance for distributed safety-critical CPS are aligned with the technical requirements as documented in D1.2. This deliverable includes a short overview of needs and expectations per concept category to summarize D1.2.

- D7 (D2.1) Reference architectures for SCDCPS v1 (M12)

  The selected concepts for end-to-end safety and performance for distributed safety-critical CPS are aligned, and ensured to be consistent, with the TRANSACT reference architecture as documented in D2.1. This deliverable includes a brief overview of this reference architecture for understanding.

- D9 (D3.2) Selection of concepts for end-to-end security and privacy for distributed CPS solutions (M12)

  The selected concepts for end-to-end safety and performance for distributed safety-critical CPS are harmonised with the complementary concepts for end-to-end security and privacy for distributed CPS solutions as documented in D3.2.

2.3 Structure of this deliverable

The structure of this deliverable is as follows. Firstly, in the next section (Section 3) a brief overview of the TRANSACT project use cases is given with a focus on each use case's safety and performance challenges, to provide context to the selected concept descriptions and associated examples for application in context of these use cases. Also, in this section included is a brief capture of the TRANSACT harmonised technical requirements, and a brief overview of the TRANSACT reference architecture. Section 4 then provides an 'helicopter view' on the structure of the concepts, i.e. three identified concept categories (application, cross-cutting, and platform), and thirteen selected concept classes, divided up across these categories.

This overview sets the scene for the actual description of selected concepts. In Section 5, the selected application concepts are described. This is followed in Sections 6 and 7 with descriptions of the selected cross-cutting concepts, and platform concepts respectively. Finally, in Section 8 conclusions are presented on this investigation and selection of concepts for safety and performance for systems in the device-edge-cloud continuum.

3 TRANSACT use cases, and the TRANSACT reference architecture

This section provides a short summary of relevant results of related TRANSACT work packages, with as purpose to help understanding the contents this deliverable. In this section, first a brief overview of the use cases within the TRANSACT project is given (from deliverable D1.1), then the TRANSACT reference architecture (from deliverable D2.1) is summarised.

3.1 Overview of Use cases (D1.1)

3.1.1 Use case 1: Remote Operations of Autonomous Vehicles for Navigating in Urban Context

In this use case, Fleetonomy and partners will develop a solution for remote control of (semi-) automated vehicles for navigating in urban environments (see Figure 1). The solution will allow vehicles to be moved from one location to another even without a driver, but with a remote operator. The operator will receive continuous feedback on vehicle state and environment, allowing him/her to assist the vehicle to navigate through urban traffic. The vehicle will have autonomy provided by current state-of-the-art automated driving solutions taking care of normal driving, and capable of detecting and reacting to arising hazardous situations.

During the TRANSACT project, the use case team will enhance the capability of the vehicle to understand its surroundings, react to pedestrian and other road user behaviours and make local decisions. The interaction and cooperation of vehicles and human operators in remote operating centre will be seamless and enhanced through visualisation and communication of the vehicle understanding and intent in augmented reality camera view and user interfaces with 3D data model of the driving environment. This allows the remote operator to have an understanding of the vehicle's independent capability to manage safe driving in a complex environment including people in different roles. The remote operator provides supervision and additional safety as well as the intelligence to resolve arising exceptional traffic situations. Hand-over of control between operator and vehicle is performed in smart way.

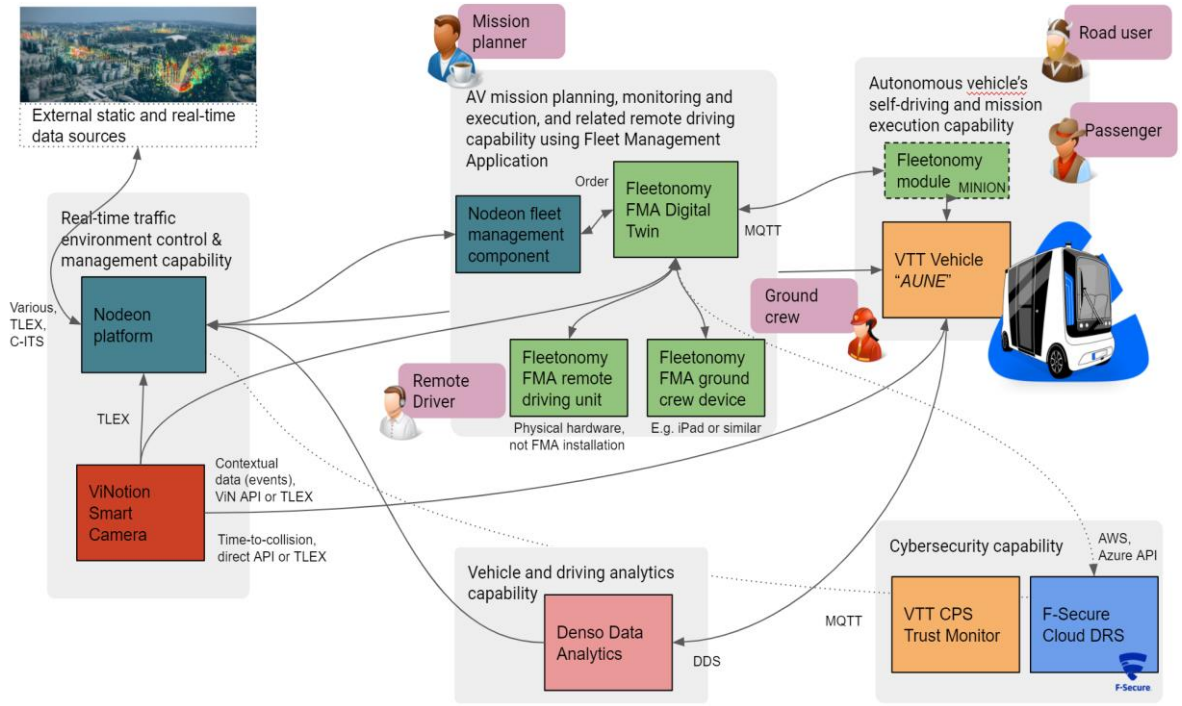

Figure 1: The remote operations use case cloud-edge-device continuum

TRANSACT safety and performance challenges. There are several performance and safety challenges, which need to be tackled within the use case. Those can be divided into 3 over-arching dimensions and addressed in 3 separate domains: the vehicle, remote operation, and the operating environment.

The first dimension is the software architecture, where the following performance, reliability and safety elements need to be addressed: vehicle autonomy, remote operating software and environmental software.

The second dimension is the network and connectivity. As depicted in the architectural diagram above, there are multiple software and hardware systems (components), which are interconnected, and most of them rely on either real-time data transfer or at least acceptable glass-to-glass or system-to-system lag where certain thresholds cannot be exceeded to ensure safe operation with acceptable performance.

The third dimension is the cloud-edge-device continuum, where much of the system performance and safety hinges on device and software integration and data flows through different systems.

The key focus in all three dimensions is on two objectives: 1) sufficient safety level and performance characteristics in the vehicle and fleet management autonomy; and 2) safe transfer of control from Remote Driver to vehicle and back.

3.1.2 Use case 2: Critical maritime decision support enhanced by distributed, AI enhanced edge and cloud solutions

The maritime use case (UC2) will demonstrate advancements in safe and efficient maritime navigation made possible by enhancing the existing basic edge/cloud technologies in the NAVTOR e-Navigation Suite to the TRANSACT architecture. This will allow for integration of traditional advisory services, AI-based advisory services and data-analytics services into the device-edge-cloud continuum to improve safety and efficiency, as will be demonstrated for automated High Sea vessels and an autonomous harbour-based support vessel.

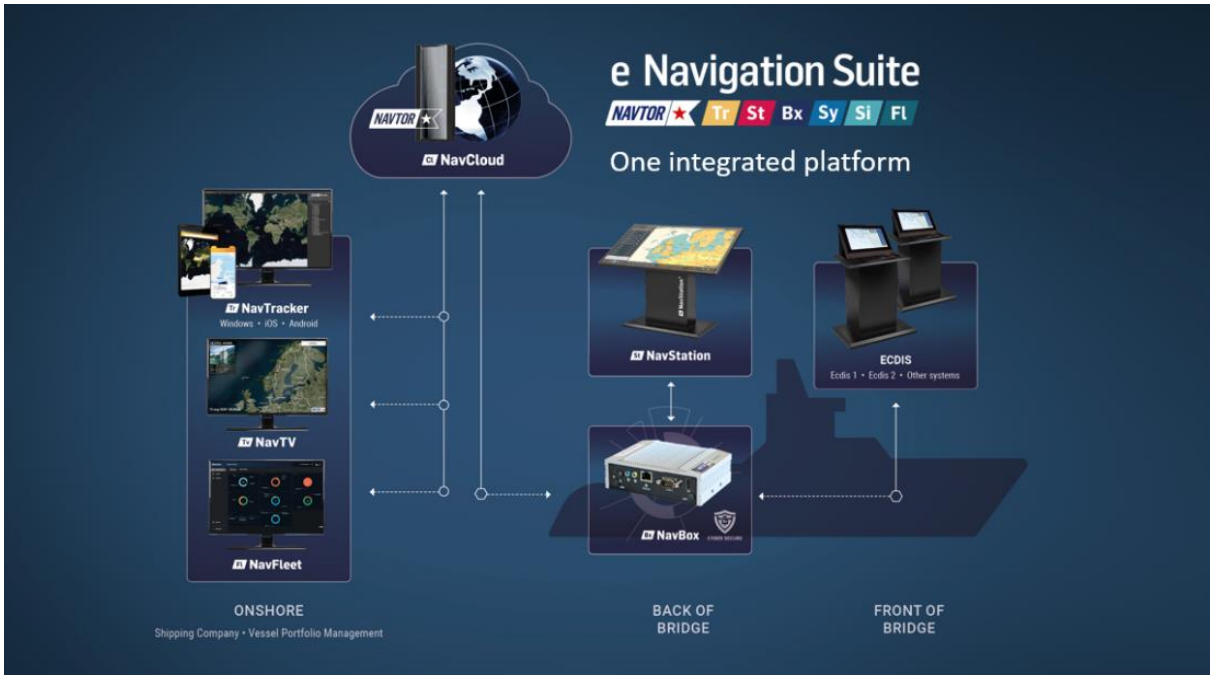

Figure 2: NAVTOR's pre-TRANSACT e-Navigation suite

In the Figure 2, NAVTOR's pre-TRANSACT e-Navigation suite is illustrated. In the following Figure 3 the planned device-edge-cloud services are detailed, building a holistic AI-based monitoring and decision support service for safe and efficient navigation.

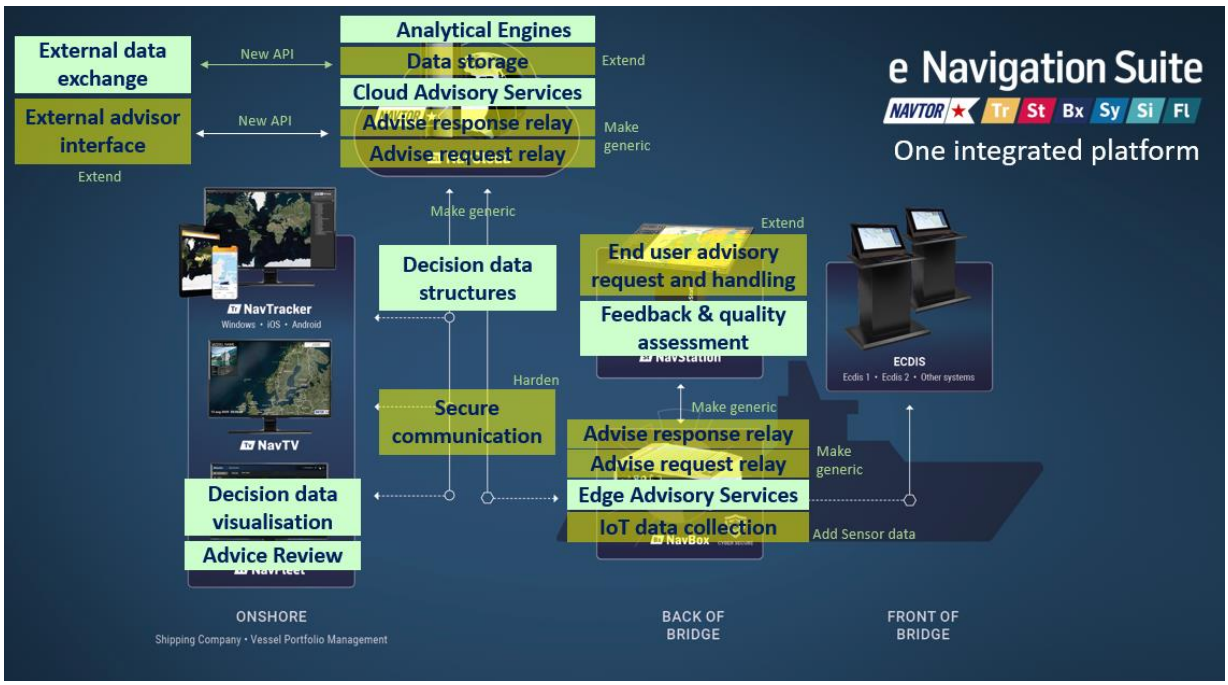

Figure 3: NAVTOR's e-Navigation suite build on TRANSACT architecture

TRANSACT safety and performance challenges. The safety and performance challenges for the High Sea demonstrator are mainly related to the safety of navigation, including the updating of critical navigational data to the Back of Bridge (e.g. the planning station) and the Front of Bridge systems (e.g. the ECDIS system). In addition SW bugs and/or human failure in operating the planning station can be critical, e.g. when making a Passage Plan. Most serious safety issues are related to the real-time monitoring and decision support for Navigators and the navigation itself by the stand-alone ECDIS system, e.g. by failing to properly monitor the traffic conditions and the surrounding traffic situation, or just because of a general lack of situational awareness.

In the demonstrator related to an unmanned surface vessel in port, safety issues are related mainly to a lack of situational awareness, which is needed to avoid incidents in due time, e.g. by stopping the autonomous vessel before such an accident could happen.

3.1.3 Use case 3: Cloud-Featured Battery Management System

In a cloud-featured battery management system, electric vehicle battery data is collected and transmitted using an advanced and secure data logger and transferred encrypted to a data broker cluster; the data is stored in an optimized database. All of this is happening while the Electric Vehicle Fleet (EVF) is driving.

Figure 4 depicts the high-level data exchange pattern between the three tiers according to the TRANSACT architecture. Safety, time and performance critical communication, control and decision-making is mainly restricted to and between the device and edge tiers. The battery management system (BMS) measures battery specific quantities and controls the system accordingly. Namely, the BMS does not require a persistent connection to the cloud in order to run and perform properly. On the other hand, the BMS is connected to the cloud via the LTE Gateway in order to send telemetric data to the cloud and receive optimization and software updates.

The realization of the architecture delivers a system of two subsystems. The device-edge subsystem, which allocates safety and performance critical elements, is based on CAN Bus communication, a well-established and proven system. The edge-cloud subsystem is less critical for safe operation and is more resilient by design against performance issues.

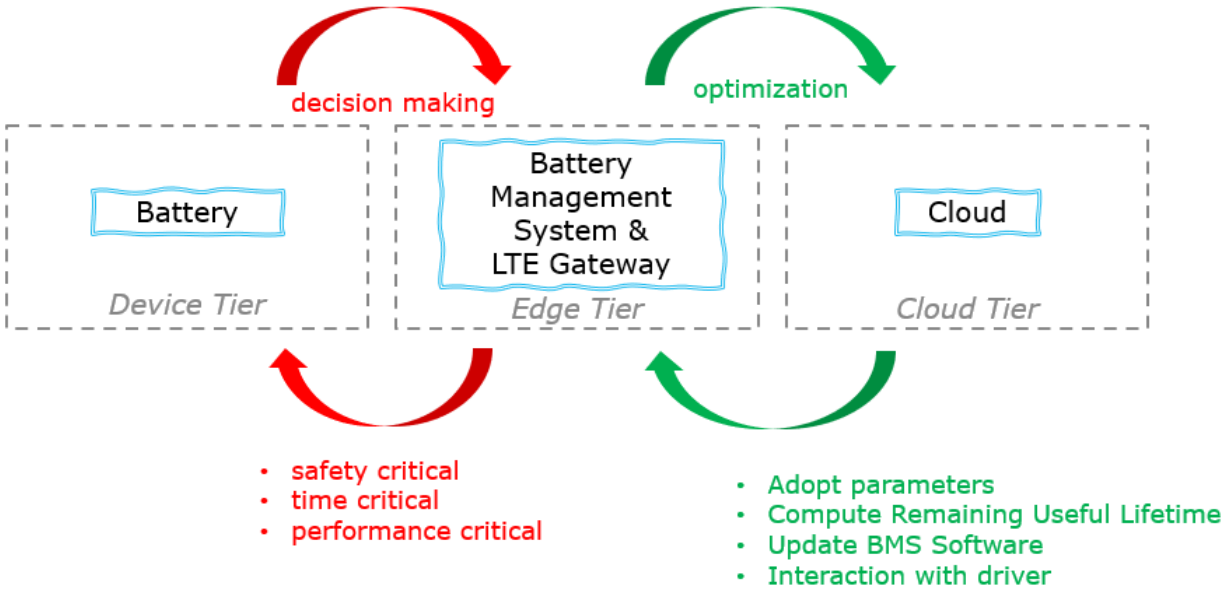

Figure 4: Three-tier structure of the cloud-featured battery management system.

TRANSACT safety and performance challenges. As shown in Figure 4, safety, time, and performance critical communication are mainly restricted to the device and edge tiers. While the communication between edge and cloud is of minor criticality regarding performance and safety, security is the critical aspect.

3.1.4 Use case 4: Edge-cloud-based clinical applications platform for Image Guided Therapy and diagnostic imaging systems

Use Case 4 "Edge-cloud-based clinical applications platform for Image Guided Therapy and diagnostic imaging systems" is a healthcare use case, aiming to improve the workflow and interoperability in hospitals. In particular, the use case addresses image based minimally invasive clinical procedures which are typically performed in Cathlabs or Operating Rooms. Clinical procedures in such environments are typically very complex and involve a team of healthcare professionals (with a variety of expertise in multiple disciplines) and many specialized devices. In order to deliver optimal treatment, all collected (multi-modal/multi-source) data should be easily available for sharing and interpretation during examination for the healthcare professionals. However, currently the exchange of information is often hampered by a lack of tools to support efficient collaboration between disciplines and often requires collecting and reviewing information in the room outside the Cathlab which slows down the performed procedure and possible effectiveness of the overall treatment.

Figure 5 Typical workflow setting during image-guided therapy with physicians utilizing medical imaging equipment for the minimally invasive treatment of patients

In UC4, a medical imaging device performing safety-critical applications, like live X-ray imaging, will remain as an embedded functionality in the device. However, mission-critical functionality, such as non-real-time image processing, offline planning and intra-operative analysis tools will be deployed outside of the device, i.e., either on the edge or in the cloud.

In addition, simulations using different system parameter settings (bandwidth, latency, system availability etc.) and real clinical workflow parameters (e.g. image reading turnaround time) will allow assessment of the impact of cloud-based components on the actual end-user workflow. These simulations will be performed even beyond the strict IGT scope and include also other imaging modalities (e.g. MRI).

TRANSACT safety and performance challenges. Safety and performance aspects are essential when complex image processing and application responsiveness come together during the live image guided treatment of patients. As an example, one scenario in UC4 is concerned with the ability to integrate pre-interventional 3D images with the live X-ray image guidance such that a physician has optimal guidance during treatment of a patient. The system must be able to perform the image registration in the cloud, within a predefined response time or revert to a well-defined degraded performance way of working.

3.1.5 Use case 5: Critical wastewater treatment decision support enhanced by distributed, AI enhanced edge and cloud solutions

Use case 5 “Critical wastewater treatment decision support enhanced by distributed, AI enhanced edge and cloud solutions” is an industrial use case, which attacks three problems: the detection upon industrial discharges, the need for better strategies for equipment maintenance and the need for a more efficient cross-WWTP operation.

Wastewater treatment plants (WWTP) aim cleaning sewage and water coming from citizen consumption, drainage and rainwater with propose of these wastewater streams can be returned safely to the environment. Sometimes, the environmental areas where the treated water is discharged are sensitive or protected areas and therefore, the correct water depuration has a strong impact in the environment, population welfare and agriculture in the surrounding zones. Therefore, *disruptions and dysfunctions* in the management of the main processes related to the achievement of proper water quality may lead to high risks to the society, the environment and the local economies. The most extended kind of WWTPs involve physic-chemical treatment and biological treatment in different stages for removing solids and pollutants, breaking down organic matter and restoring the oxygen content of treated water (see Figure 6).

Unfortunately, those disruptions on the depuration processes usually happen, especially in industrial areas, where the WWTP are severely affected when the toxic spills reach the facilities, leading to an interruption in the operation of the critical biological reactors, avoiding an appropriate water depuration. Therefore, these toxic discharges have impact on environment and could seriously affect the protected natural area (fish kills). The re-establishment of each biological reactor may involve around 20k-25k euros, aside from the heavy penalties for the plant managers.

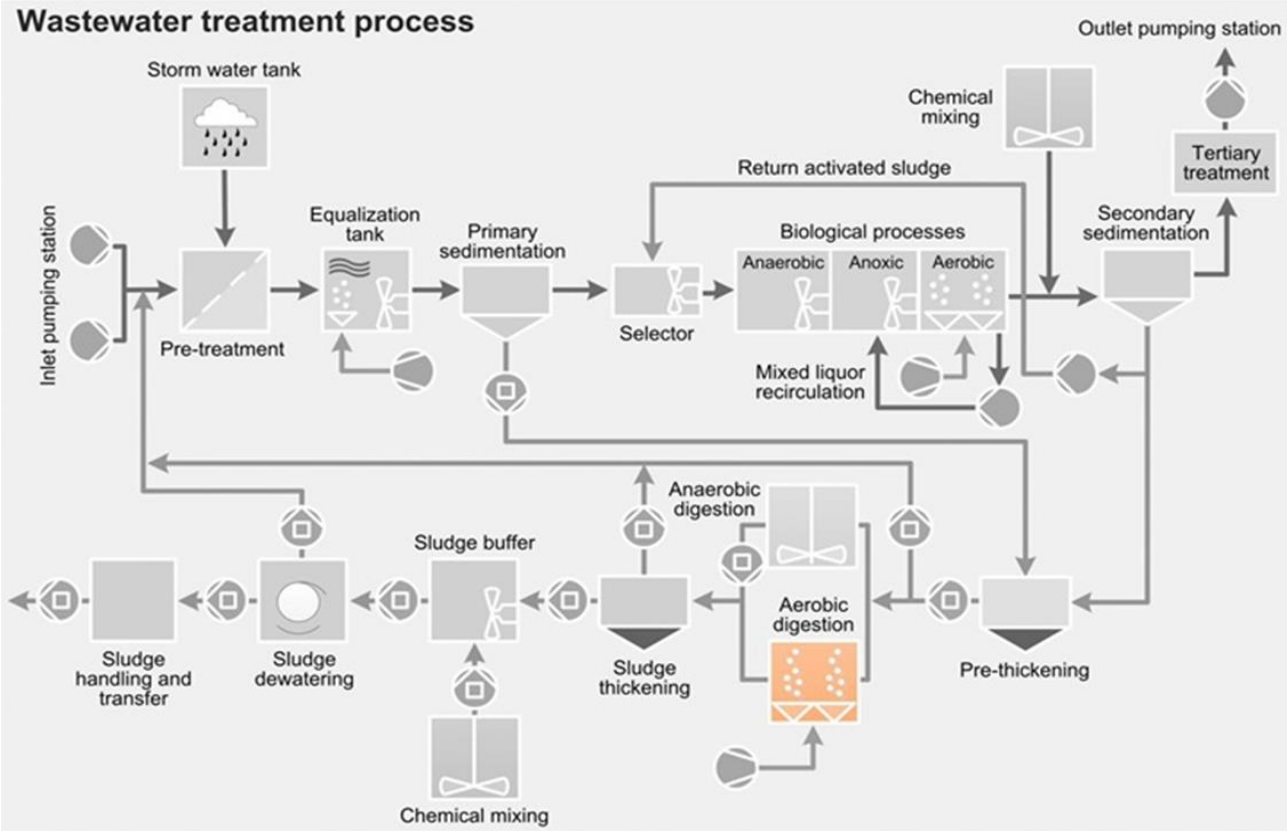

Figure 6 General scheme of a typical wastewater treatment plan process

TRANSACT safety and performance challenges. The safety challenges in this use case are mainly related to the detection of these toxic spills and the prevention of damage to facilities. Monitoring and modelling these wastewater treatment processes and detecting anomalies, upcoming discharges could be predicted, advising the operators or automating the operation in certain situations (e.g., closing the entrance to one of the reactors). Moreover, WWTPs work as isolated islands, sharing information in real time across WWTPs may help other WWTPs downstream to prepare for upcoming spills.

3.1.6 TRANSACT Horizontal demonstrator

In this demonstrator, most of the core TRANSACT components will be deployed showcasing them and allowing to evaluate the correctness and generality of the architecture proposed in the project. Additionally, this demonstrator aims to integrate and evaluate the mechanisms and components developed in WP3 related to safety, performance, security and privacy. The integration of these components will be evaluated through a generic scenario that will be used as reference to combine the different core and functional building blocks using the TRANSACT approach. This demonstrator will also assist in the deployment and maintenance of the software modules required at the edge and cloud levels of the use cases.

TRANSACT safety and performance challenges. The challenges in this demonstrator will be mainly focused on operational mode changes, data and communication and monitoring through the continuum cloud, edge and device tiers. This demonstrator checks aspects related to reliability, availability, safety, confidentiality and integrity in a general TRANSACT platform.

3.2 The TRANSACT reference architecture (D2.1)

The TRANSACT project has adopted a tree-tier, device-edge-cloud, architecture concept. Based on this concept, the project has proposed a first reference architecture in deliverable D2.1 (see Figure 7). In the deliverable D2.1 a full description is given of the TRANSACT reference architecture; here a summary is included for positioning the selected safety and performance concepts in this reference architecture.

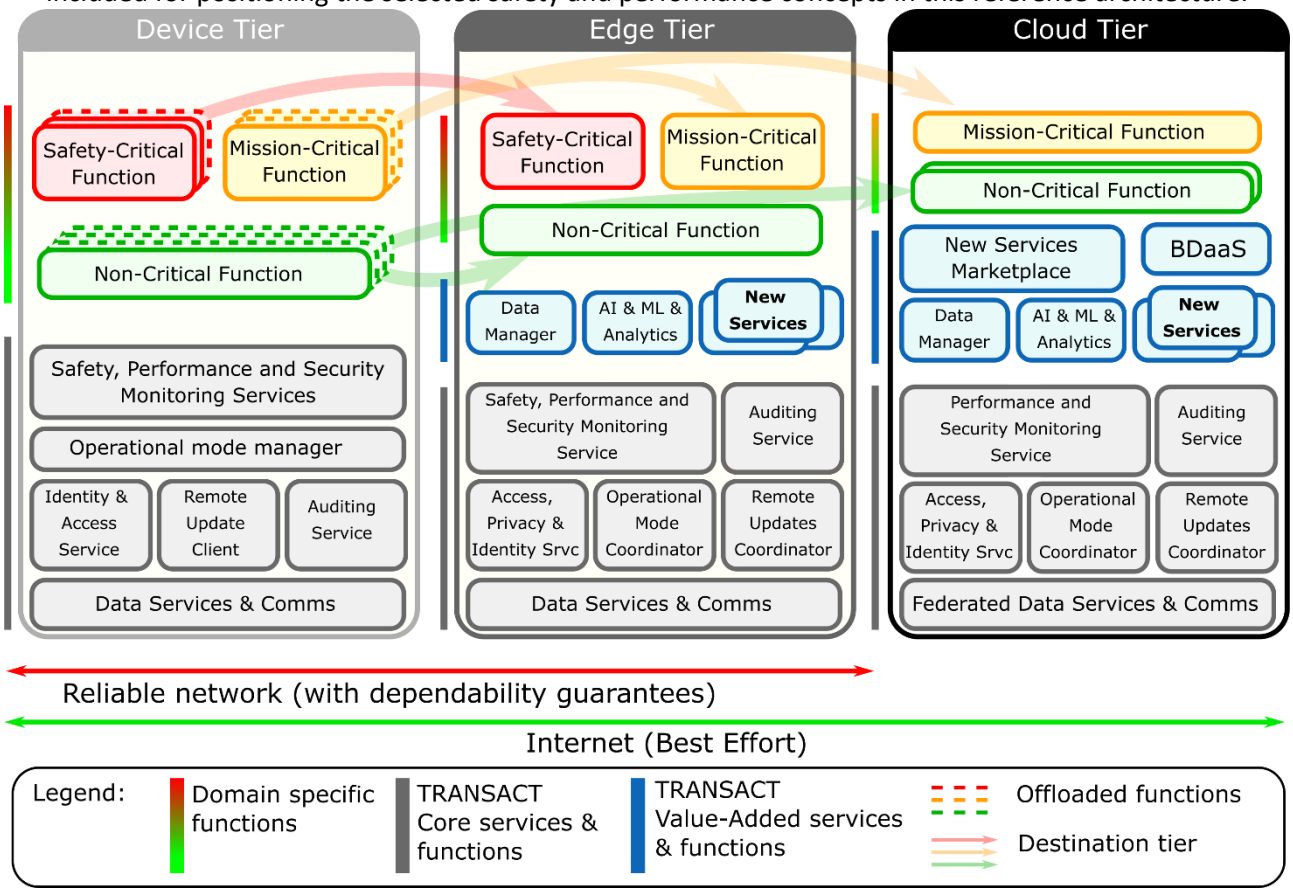

Figure 7: TRANSACT reference architecture

The domain specific functions or components are depicted in red, yellow and green depending on their criticality. Domain specific functions may be offloaded from the device to other tiers. Core TRANSACT components, available to every use case, are depicted in grey. Finally, blue components refer to potential Value-Added functions that may be included depending on the use case.

The TRANSACT reference architecture defines the safety and mission critical functions, the core services and functions, and further value-added services and functions (see Figure 7).

The safety and mission critical functions are key in the safety-critical CPS. The distributed safety-critical CPS solutions will be deployed over 3-tier (device-edge-cloud) architecture continuum. Each tier in the

architecture provides a specific quality-of-service level especially with respect to performance aspect such as response times and data-transfer guarantees, ranging from best effort to reliable and time-deterministic data transfers. Safety-critical functions often have hard real-time constraints, whereas the mission-critical functions may have soft real-time constraints (which may degrade the system's quality of service when missed, but do not necessarily lead to failures). In edge and cloud, value added functions can be deployed including in the cloud Big Data as a Service (BDaaS) services.

TRANSACT aims at improving over monolithic CPS by offloading functions to the edge or cloud tier. A few use cases will offload safety-critical functions to the edge tier, more use cases will offload mission-critical functions to edge and cloud. Such offloading gives numerous advantages such as: improved reliability and performance of the device (as fewer services are running on the device), improved efficiency of the offloaded functions due to usage of better hardware in the edge or cloud, improved innovation speed of the distributed CPS as the new or upgraded functions can be deployed easier in the edge and cloud.

However, when considering offloading functions from the device it is critical to ensure CPS system end-to-end safety, performance, security, and privacy. Therefore, a number of dedicated core services are introduced to cooperatively realize that objective. The *safety, performance and security monitoring services* are responsible for monitoring, detecting and preventing safety, security and performance failures. In addition, they track the devices' KPIs (such as, latency, throughput, accuracy, availability) that are used by the *operational mode manager* (running on the device) and the *operational mode coordinator* (running at the edge/cloud tier) to decide at runtime whether a device's function can be executed remotely or not.

To achieve safe and predictable updates to the system the following core services have been identified: the *remote update client* (running on the device) and the *update coordinator* (running at the edge/cloud). Those services cooperate across tiers to perform remote automatic updates of the different device services in a secure and safe way. Each update activity is coordinated with the operational mode coordinator service to keep the system in the safe state at any time.

Further core services address additional security and privacy concerns. The *identity & access* services are responsible for granting/denying access to the system resources based on the policies defining who has what access (in which role) to which resources. Other core services contributing to the system security are the *auditing* services. These services collect information about accessing and using the system in order to help detecting the security policy violations when system is accessed by unauthorized users or in an unauthorized way. The security aspects are also addressed by the *(federated) data services and comms* services helping in efficient and secured data handling, both, in transit and at rest

In this project, each use case will experiment with the TRANSACT reference architecture, its components, and a mix of the selected concepts presented in this deliverable with the aim to capture the overarching results across the various use cases. This allows TRANSACT to validate the approach and refine the proposed reference architecture over the course of the project.

4 Overview of selected concepts for safety and performance

4.1 Introduction

In the course of the investigation in task 3.1, an overall structure was created to organise the selected concepts and maintain overview. This section presents a brief overview of this structure with three categories and thirteen concept classes.

4.2 Overview of selected safety and performance concepts

In systems design, one often distinguishes the platform (the infrastructure of a system) from the applications that run on this platform, or are enabled by this platform. The concept categorisation follows this distinction. The selected concepts are thus divided into three categories:

- Application concepts
- Cross-cutting concepts
- Platform concepts

Application concepts target the Application: the functionality that implements a particular solution to help the end-user to perform a specific task. An application can be composed of a monolithic service or a group of distributed services which are executed in different and distributed targets in the device, edge, cloud continuum.

Platform concepts target the Platform: the environment in which the application is executed. It comprises of the complete infrastructure in the device, edge, cloud continuum to execute an application, including hardware, operating systems, hypervisors, communication networks, containers & cloud computing services.

Cross-cutting concepts are system-level methods and techniques for linking application and platform. They include concepts for designing and deploying the application on the platform, as well as analysing and run-time monitoring and managing the behaviour.

The selected concept classes, based on these three categories, are shown in Figure 8.

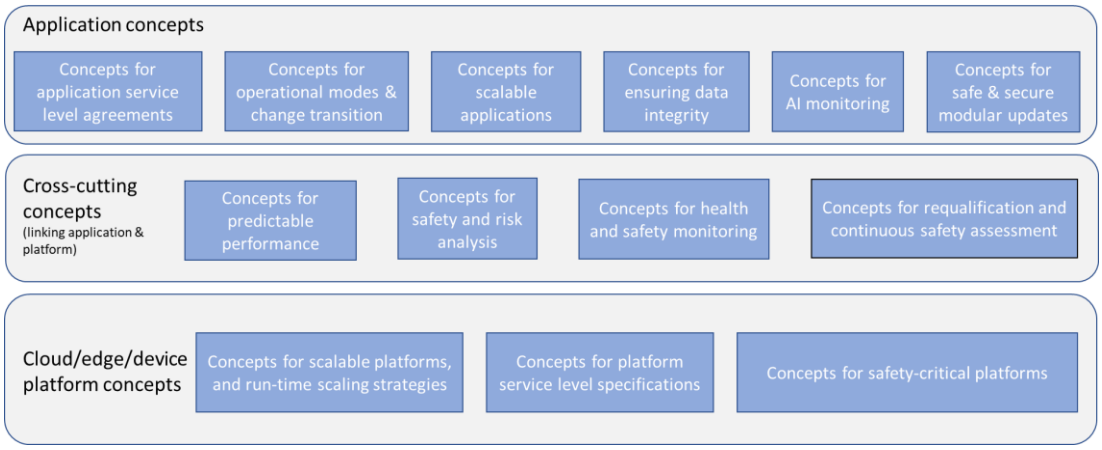

Figure 8: Selected safety and performance concepts

A concept class then may contain a small number of related concepts as needed to cater to the needs of various use cases and their domain characteristics.

In the next sections, the selected concepts are described per concept category (Application, Cross-cutting, Platform) and concept class (e.g. predictable performance, safety and risk analysis, or AI monitoring).

5 Application concepts for safety and performance

5.1 Introduction

This section describes the selected application concepts.

The Application is the functionality that implements a particular solution to help the end-user to perform a specific task. An application can be composed of a monolithic service or a group of distributed services which are executed in different and distributed targets in the device, edge, cloud continuum.

5.2 TRANSACT project harmonised needs and expectations

Following the TRANSACT reference architecture (see section 3.2), use case applications will be divided over the device-edge-cloud continuum. For safety-critical and mission-critical functions, this means that additional precautions are required. On top, further added-value and AI functions need to be integrated.

For such distributed applications, this means first and foremost that service level expectations need to be formalised, monitored and assured to be met. For AI services in particular, concepts for such service level expectations, monitoring and management of AI services need to be developed to fit with use with/in distributed safety-critical CPS solutions. As edge and cloud in general have no absolute availability guarantees, on-device fallback functionality and seamless change-over handling to fallback functionality need to be managed, so that at all times safety is warranted. This requires applications to offer various operating modes, be scalable, and support seamless change of operation modes. In distributed safety-critical CPS solutions, safety and privacy related data leaves the confines of the device, and vice versa external data is imported for use in safety-critical or mission-critical functions. This requires the data integrity to be established and safe-guarded. Finally, to reap the benefits of the cloud, it needs to be possible to apply incremental updates of functionality across the device-edge-cloud continuum in a safe and secure manner.

The complete set of technical requirements is documented in TRANSACT deliverable D1.2. This section describes selected application concepts that address these requirements and lists open challenges for these concepts as applicable that will be investigated during the TRANSACT project.

5.3 Selected application concepts

The selected application concepts are highlighted in Figure 9.

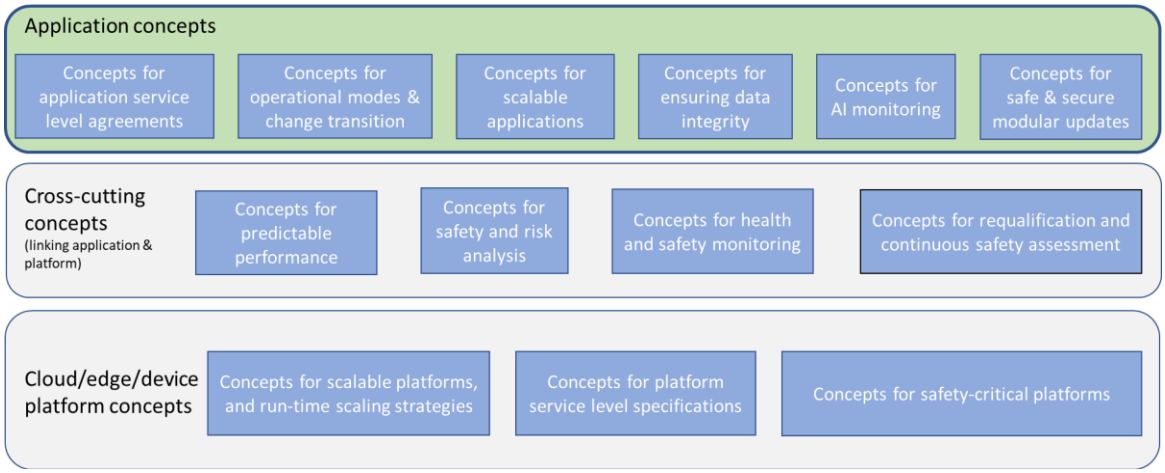

Figure 9: Selected application concepts

5.4 Concepts for application service level agreements

This section describes concepts and methods for the specification of safety and performance at service level. A service-level agreement (SLA) is a document that defines the level of service expected by a customer from a supplier. SLAs (Stantchev & Schröpfer, 2009) specify the metrics (indicators for defining Service Level Objectives – SLO) by which that service is measured and the penalties and remedies if the agreed-upon service level objectives are not achieved.

A well-defined SLA will contain the following components:

- Type of service to be provided: It specifies the type of service and any additional details of type of service to be provided.
- The service's desired performance level, especially its reliability and responsiveness: It specifies the level of reliability and responsiveness of the service. A reliable service will be the one that suffers minimum disruption in a specific amount of time and is available at almost all times.
- Monitoring process and service level reporting: This component describes how the performance levels are supervised and monitored. This process involves gathering different type of statistics, how frequently these statistics will be collected and how they will be accessed by the customers.
- The steps for reporting issues with the service: This component will specify the contact details (of a person or organisation) to report the problem to and the order in which details about the issue have to be reported. The contract will also include a time range in which the problem will be looked into and when the issue will be resolved.
- Response and issue resolution timeframe: Response timeframe is the time period by which the service provider will start the investigation of the issue. Issue resolution timeframe is the time period by which the current service issue will be resolved and fixed.
- Repercussions for service provider not meeting its commitment: If the provider is not able to meet the requirements as stated in SLA then service provider will have to face consequences. These consequences may include customer's right to terminate the contract or ask for a refund for losses incurred by the customer due to failure of service.

SLA elements could be divided into Service and Management elements. Service elements include the specifics of the services provided. Management elements should provide information about

- measurement methods and standards,
- dispute resolution processes,
- and reporting processes.

Standard types of metrics could be used for the SLA monitoring:

- Service availability: This is the amount of time the service is available for use, with e.g. target of 99.999% availability (‘five nines’), where availability is specific for the service characteristics, e.g., accessibility of data and retention periods.
- Business metrics: A business metric is a specified measure used to track the status of a specified business process, e.g. a security-breach notification may specify that a client is notified within 48 hours after determining the existence of a security data breach.
- Technical metrics: Technical metrics are used to quantify and assess the critical technical attributes of delivered services. The main purpose of the technical metrics is to detect and fix issues before they impact the SLA e.g. a specific service may specify a response time of 10 seconds for 99.99%. of requests.

5.4.1 Application service level agreements for AI-model based distributed CPS solutions

Overview

For distributed CPS solutions that run on top of AI models, SLA elements are tightly bound to AI services and the monitoring of such AI services (see Section 5.8: Concepts for AI monitoring). Monitoring process and service level reporting have to specify both business and technical metrics. Some AI monitoring metrics could be used directly, or modified, for use as SLA metrics.

By design, the CPS systems usually use a full spectrum of cloud services: IaaS, PaaS, and SaaS. In this section, the main focus is on AI-related SLA procedures, therefore the proposed concept is based on SaaS. For systems and applications that implement and use AI/ML models, specific measurement methods and standards are still not well established. AI and machine-learning models degrade in terms of performance over time. They are dynamic and sensitive to real changes in the real world. Drift or new pattern of distribution in data could affect models' performance and quality.

For example, in maritime use case (UC2), AI models are expected to be used for producing good routing advices. The AI models that produce routes candidates are trained on historical data and implemented in advance as components of the CPS solution. SLA measurements and procedures will be used to guarantee that models will be re-trained and updated in case of models' deterioration.

For systems and applications that use AI/ML models, specific measurement methods and standards are still not well established. AI and machine-learning models degrade in terms of performance over time. They are dynamic and sensitive to real changes in the real world. Drift or new patterns of distribution in data could affect the models' performance and quality.

SLA should define procedures for maintenance of deployed AI/ML models during the whole model lifecycle.

For AI/ML models, main metrics could be described in following way:

Service availability

Models that are included in systems/applications are functional during all the productive time. It means that models are giving answers/results as expected and having a delay that does not exceed threshold value mentioned in SLA.

Business metrics

Achieving an agreed level of operational performance and uptime greatly relies on minimizing the average time between AI system breakdowns or mean-time-between-failure (MTBF). Solution availability in a production environment could be expressed as a percentage of uptime each year. Due to requirement that a CPS has to operate continuously, service level agreements (SLAs) should refer to monthly or yearly downtime or availability in order to calculate service credits in proportion to the corresponding period of time that the system was unavailable.

Quality of model answers/results should not degrade over time. For example, metrics could be built based on user reactions (quantity of accepted/ignored advices, decrease of accepted advices). Such metrics could be produced by AI/ML monitoring systems and concepts (see also Section 5.8: Concepts for AI monitoring).

Technical metrics

SLA measurements are a key component of any successful AI-driven system in a real-life business context.

There are many AI-related technical metrics are potentially useful in an SLA, e.g. as follows:

- Classical Accuracy/Precision/ Recall and similar metrics.

- Monitoring model predictions over time and compare the distribution using statistical metrics such as Hellinger Distance, Kullback-Leibler Divergence. Both Hellinger Distance and Kullback-Leibler divergence methods are used to quantify the similarity between two probability distributions. It can help to discover drift of model predictions.
- CPU/GPU utilization when the model is computing predictions on incoming data from each API call; how much model is consuming per request.
- Memory utilization for when the model caches data or input data is cached in memory for faster I/O performance.
- Number of failed requests by an event/operation.
- Total number of API calls.
- Response time of the model server or prediction service.
- Data quality metrics – tracking the statistical metrics according to the data that flows in. This could be basic statistical properties of the data such as mean, standard deviation, correlation, and so on, or distance metrics (such as KL divergence, Kolmogorov-Smirnov statistic).
- The actual time for retraining job during the model lifecycle, including details about time used by the retraining job to run, resources usage of the job, and the state of the job (successfully retrained or failed).

Fit with concept TRANSACT reference architecture/components

In Figure 10, the light blue ellipses highlight areas which should be used or covered for measuring and monitoring an SLA.

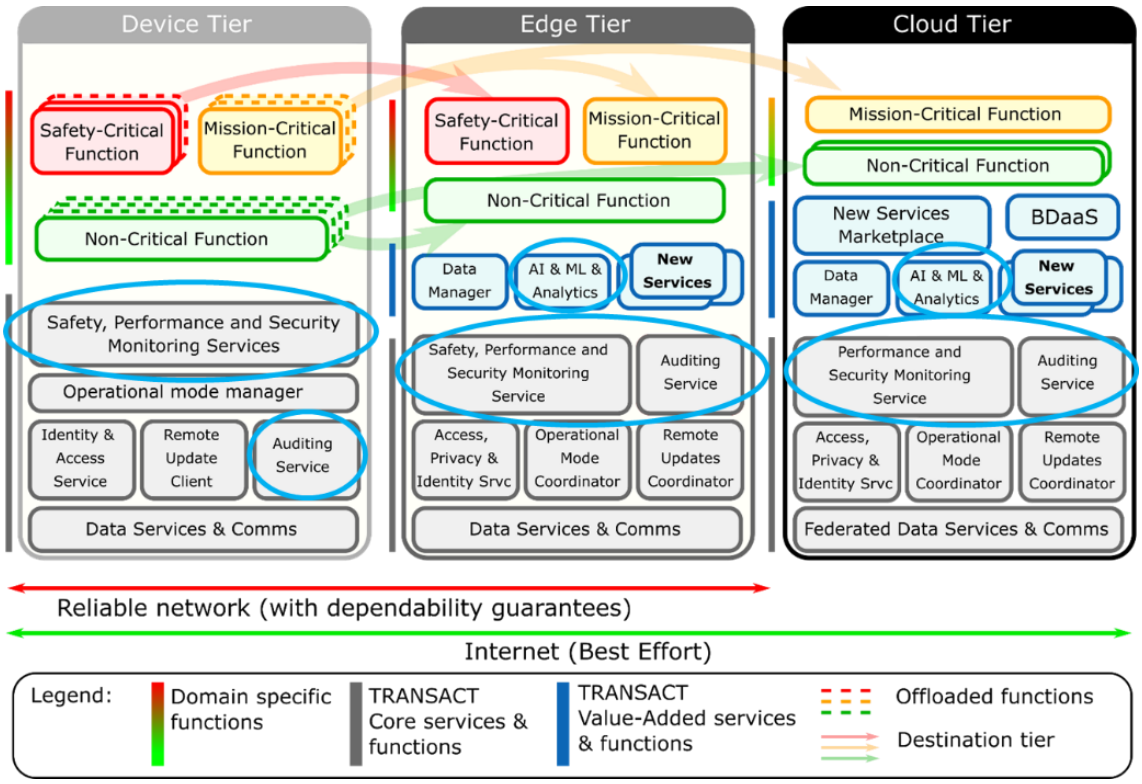

Figure 10: TRANSACT reference architecture

Example in context of a use case

We describe the concept using the maritime use case (UC2). In order to guarantee the quality of the TRANSACT AI-based solution during the whole lifecycle, the solution should be supported by a service provider based on the SLA. The SLA should use results from monitoring and audit to achieve an agreed level, by e.g. agreed KPIs for business metrics, of operational performance and uptime.

For example, increase in quantity of routing advices ignored by a Navigator should be reported to the service provider and issue a re-training of AI-models. The following scenario illustrates this problem: In the beginning, the AI model produces good routing advices. It is newly trained and proposes quite efficient and safe routes between various ports. Due to the good quality of the advices, the routes produced by the AI model are often accepted by the Navigator. After some period of time however, the model and quality of resultant proposed routes start to degrade. The Navigator rejects advices more often due to their inefficiency or lack of safety awareness. Metrics in the context of an SLA, and based on the quantity of ignored routing advices, report the problem to the provider, who then can take action.

Challenge for application within TRANSACT context

It is still unclear how to decide when AI-models need re-training or calibration of parameters and how maintenance activity should be performed “on the fly” (preferably without affecting the performance of the solution). Suitable re-training strategies will be investigated in the further course of the TRANSACT project, notable in task T3.3.

Another challenge is a data drift caused by use of the solution. Data distribution will be changed due to more optimal way of navigation and fuel consumption. It will affect quality of AI-model that were trained on non-optimal or sub-optimal data.

5.5 Concepts for operational modes and change transition

Systems often exhibit multiple behaviours during its execution lifetime. These differentiated behaviours are implemented using operational modes, each of which is characterized by its own set of functionalities, i.e., its set of tasks. At any time instant, only one of the different modes is selected, which is known as active mode. The modes are defined at the application layer, since they are an integral part of the executing system. The application handles the information of when and how a change in the current active mode is required. Such systems are known as multi-mode systems (Burns, 2014). Several examples of multi-mode systems can be found in TRANSACT use cases. As an illustration, the remote driving use case (UC1), where at least two operational modes can be found: remote driver and local driver.

Switching from the current mode (also referred to as old-mode) to another one (the new-mode hereafter) requires substituting the current executing tasks with those of the target mode. This process introduces a transient unstable stage, where tasks of both old and new mode may coexist. This leads to an overload which may compromise the system schedulability (i.e., the ability to meet all timing constraints of the system). This is highly undesirable in real-time systems, where the correctness of the system not only depends on producing correct output but also on providing it at the right time. This is even more complex in distributed CPS solutions, where some components may, at some point, still be unaware of a mode transition being in place due to messages not having arrived yet. Coordination between the components is required in order to safely perform a mode change.

As seen, designing a distributed multi-mode system is complex and challenging. Schedulability must be ensured at each different operational mode and in each possible transition between modes.

5.5.1 Operational mode taxonomy

Modes in multi-mode systems can be accessed either through a statically defined sequence or through different non-predictable run-time events.

Three types of modes can be generally identified, which partitions the functionality of a system naturally:

1. We call *Normal Functional Modes* to the different planned phases the application moves through regularly. An example of such modes would be the different phases aircraft flights progress (e.g., taxiing, take-off, level flight, etc.).
2. *Exceptional Functional Modes* are phases the system must go through after rare events happen. These events cause code to be executed that would otherwise not be required. This response is planned in the design, but this mode change may never occur during the execution of the system. An example would be the 'prepare for crash' mode that can be found within a car. When the monitoring system detects that an impact is imminent, different measures must be taken (e.g., seat belt tightening, deploying the airbags, etc.) that would normally not happen.
3. *Degraded Functional Modes*, oftentimes also called *Graceful Degradation*, are those modes where for unsafe circumstances (e.g., errors or erroneous system states) the system load must be shed, and priority must be given to issues of safety and minimum functionality to avoid catastrophic consequences. The general response to these mode change events is planned, but the full set of error conditions that can trigger this change may not be known in advance.

Each mode has its own set of tasks. Therefore, a mode change can have different implications depending on their respective task sets. For instance, a task may run in both modes unaffected by the mode change, or it may run only in either the old or the new one. Additionally, tasks may change their parameters depending on the mode it is being executed on. For example, in this latter case, a task can have its period, deadline, worst-case execution time or even priority scheme altered. It falls on the transition protocol to decide how the tasks are affected, i.e., which tasks must be aborted/activated/altered.

The transition between two modes is initiated by a mode change event, sometimes also called request or trigger. Usually, this trigger is related to the current state of the system or to its environment. For example, a time-triggered event can be coordinated to the time of the environment, hence having systems that could switch between night and day modes. Other triggers may come from the hardware platform sensors or failure modes or some health monitoring subsystem. This is what graceful degradation usually employs.

After a mode change request is issued, a mode change protocol is required to manage the actual mode change. As already mentioned, the mode change protocol is in charge of handling the changes in the tasks being executed during the unstable period between old and new mode. The definition of the mode change protocol must be closely tied to the form of analysis used to verify the schedulability of the system during these transition periods. In the next section the more common protocols will be explained, while also providing some more recent and refined techniques in the literature.

5.5.2 Mode change techniques

Overview

The TRANSACT reference architecture proposes the management of the execution of the different safety-, mission- and non-critical functions across the three-tiers architecture. The tasks associated to the functions are identified at design time and deployed in the required tiers. Each of the possible configurations on where to execute these tasks translate into the functional modes described in section 5.5.1. For instance, one of such scenarios is the run-time migration of safety- and mission-critical functions from the device to the edge tier. This transition involves, among other actions, the activation/deactivation of tasks in the different tiers,

according to the plan of the new mode. A protocol must be defined to handle this transition and ensure compliance with safety requirements. Such protocols can be characterised typically in three ways:

- *Immediate:* The request event causes the mode change to take effect immediately. Jobs of the old mode are suspended or aborted and jobs from the new one can start.
- *Bounded:* The system remains in the old mode until all released jobs have finished its execution, but no new jobs are issued. Once there are no more active jobs from the old mode, the mode change happens and the jobs from the new mode can start. This happens within a bounded time, hence the name. Oftentimes these protocols are also called synchronous.
- *Phased:* After a mode change request is issued, the old mode jobs are allowed to complete, and the new mode jobs are started after a certain time defined at design-time. Oftentimes these protocols are also called asynchronous.

With Immediate and Bounded protocols, the system is only ever in one mode, while with Phased there is a time period where both modes may overlap, thus jobs from both modes coexist. This is why phased changes are more difficult to analyse, since the load of the system is higher during the transitions than in their stable modes. For that reason, a lot of research effort has been put in evaluating how to obtain the most efficient time delay in asynchronous protocols. Also, overlapping is also inevitable in distributed or multiprocessor systems. A phased change is required as it is not possible to simultaneously inform the entire system of the need of a mode change.

It is difficult to select which mode change protocol is the best for a given system. In SafeMC (Chen, 2018) a syntax is proposed that can be used to express mode change protocols, both traditional and hybrids. Furthermore, they present a system called SafeMC that implements those language primitives and that allows the evaluation of the currently defined protocol. Since applications typically require customized mode change protocols, adopting a syntax such as the one previously mentioned during the design process, can ease the evaluation of the protocol, as well as allow for a form of standardization between partners.

5.5.3 Operational mode schedulability analysis

Overview

In safety critical systems, all timing constraints must be satisfied. From a schedulability point of view, each mode can have different requirements. As a consequence, schedulability in one mode does not imply neither schedulability in another mode nor in any of the phased changes. For that, all modes and all phased transitions must be analysed. This is crucially important in distributed systems, where the different nodes require of communication messages to notify the mode requests. This delay introduced by the network needs to be taken into account by the analysis methods employed. In that regard, some studies (Azim, 2014) suggest generating real-time schedules for the efficient utilization of the available bandwidth in the communications between distributed components.

The Response Time Analysis (RTA) technique is one of the many analysis techniques that have been developed to assure that no deadline misses occur in a single-mode set of tasks. One of its main benefits is that it can be applied to many scheduling algorithms such as earliest deadline first (EDF) and fixed-priority (FP). It can also be used for testing multiprocessor systems, specially at the device tier. RTA is based on the notion of interference between tasks, which is defined as the cumulative time that a job, which is ready to execute, cannot execute due to the execution of other higher-priority jobs.

Recently, an extension of the RTA framework is proposed (Baek, 2020) that allows the calculation of interference for tasks in the presence of a mode transition. This framework can be used, for instance, to evaluate the safety of a multimode system that has been designed with the syntax primitives described in

SafeMC. As such, using both can greatly benefit the design process, mainly in regards to the evaluation of the real-time constraints of the application.

5.5.4 Operational mode run-time management

Overview

Previous sections tackled the early stages of multimode system development, i.e., design and analysis. The run-time management of the operational modes is highly dependent to the actual software implementation. Typically, there are two elements involved in mode changes: the *mode manager* and the *mode changer*.

The *mode manager* detects the conditions that should trigger a transition and initiates this process. This component can follow two distinct approaches. In a centralized approach, the mode manager is a separate entity that receives all relevant system events and that decides when to proceed with a transition process. On the other hand, in a distributed approach, the tasks are the ones responsible of detecting the conditions and of triggering the corresponding mode change request.

The *mode changer* is responsible for performing the update process of the attributes of the tasks involved in a given change. As before, it can follow either a distributed or a centralized approach. In a distributed one, each task is responsible of changing its own attributes for the new mode after a request of the mode changer. On the other hand, in a centralized approach, it is a separate entity that can update the attributes of the involved tasks directly.

Saez et al (Sáez, 2012) proposed a framework for multiprocessor, multimoded real-time application in ADA. They model the different system modes and transitions using UML finite state machines (FSM). Using that as an input for code generation, they obtain a basic implementation for a centralized mode manager and a distributed mode changer in the ADA language.

5.5.5 Operational modes and change transition in TRANSACT

Fit with concept TRANSACT reference architecture/components

In Figure 11 the red ellipses highlight the components in the three tiers of the TRANSACT reference architecture that are responsible for the operational modes and their transitions. The *Operational Mode Manager/Coordinator* components decide when it is the appropriate moment to trigger a mode change request and ensure these changes meet safety and performance requirements. This decision is based on the parameters provided by the *Safety, Performance and Security Monitoring* components, which measure indicators from the internal state of the application and from the environment where the application is being executed in. The implementation of the mode manager/changer, described in section 5.5.4, inside the *Operational Mode Manager/Coordinator* components may vary depending on which of the three tiers the decisions are taking place.

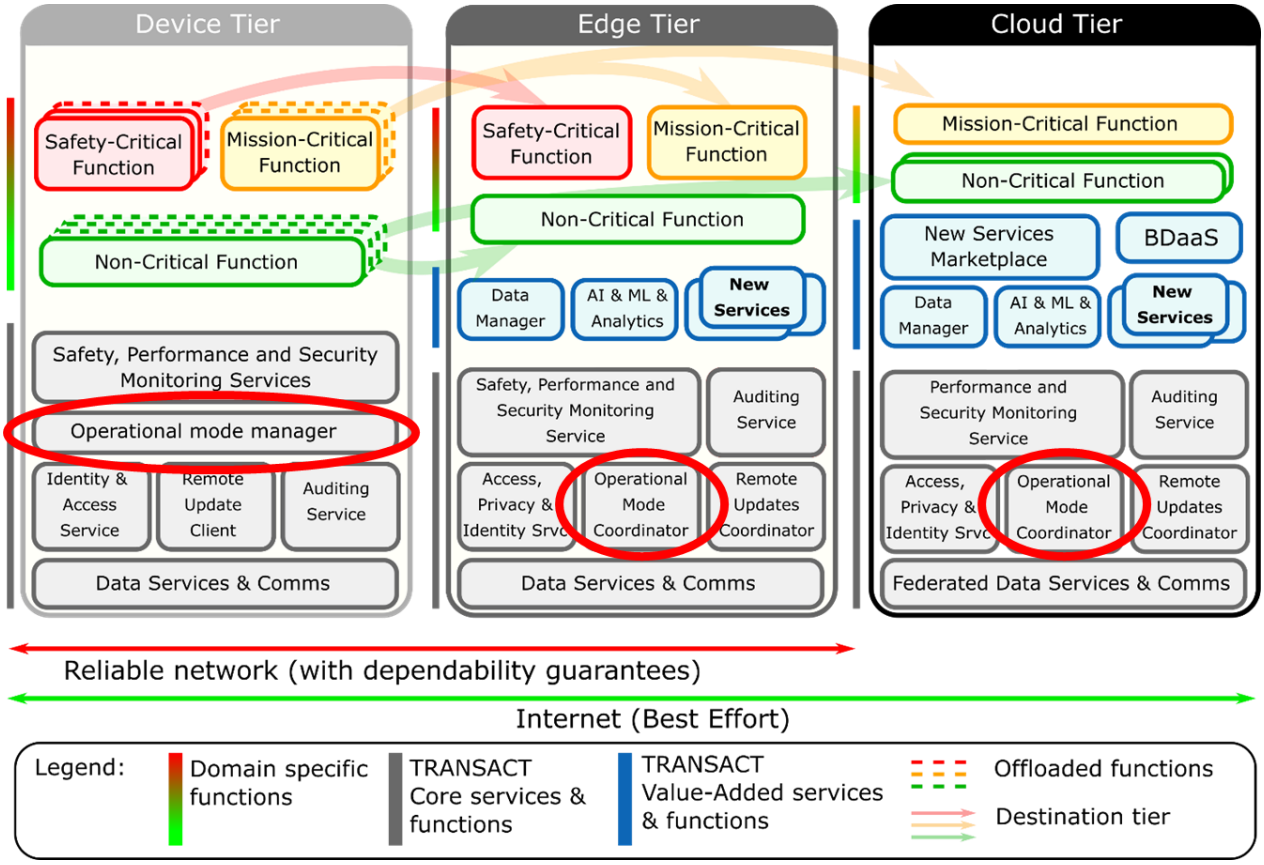

Figure 11: TRANSACT reference architecture

Example in context of a use case

An example of the abovementioned concept for operational mode change can be seen in UC4 about Image Guided Therapy. In one particular clinical scenario in this Use Case, live 2D X-ray imaging during patient treatment is combined with 3D image data that has been acquired before the procedure and is stored in the cloud. The live X-ray images are uploaded to the cloud as well where an image registration algorithm transforms the 3D dataset in such a way that the 3D image data can be correctly overlaid on top of the live 2D X-ray images.

The uploading of the data, the 2D-3D image registration and the downloading of the result must happen within a predefined response time since it is part of an interventional procedure where a patient is receiving treatment. If, for some reason, the response time cannot be met (e.g., due to unavailability of network, limited bandwidth) the IGT system must recognize this situation and switch to another operational mode in which either a fall-back solution is provided based on an on-premise compute platform or user guidance is provided that the image registration should be done manually.

Challenge for application within TRANSACT context

The operational modes and change transitions in TRANSACT require:

- Identification of the operational modes, the different available transitions between them and the triggers that enable the mode change.
- Appropriate mode change protocol selection for the coordination of the transition of all distributed components.

- Adequate decision making. Global decisions may be taken at edge or cloud level. Meanwhile, there can also be local decisions taken at device level (e.g., due to connectivity changes).
- Transitions at device/edge level must comply with temporal constraints to ensure the valid execution of safety critical functions. Modelling and analysis must be thoroughly performed to guarantee that the constraints are satisfied.

5.6 Quality-of-Service concept for scalable applications

Scalability of applications is a one of the key enablers for operational mode management and change transition (see Section 5.5). Scalability of applications can be achieved by a trade-off of an application's Quality-of-Service (QoS) level and the resources available to the application. This trade-off becomes necessary when the available resources, e.g., processing resources, bandwidth, or memory, are not enough to let the application meet its timing constraints when running at a certain QoS level. Having such a scaling capability is particularly important for systems with real-time requirements that execute tasks with varying complexity and hence varying resource allocation.

If sufficient resources are available, real-time performance can be guaranteed. For distributed systems that are considered in TRANSACT, this could mean redistributing the tasks over the devices, edge and cloud resources to exploit all resources optimally. However, due to the heterogenous nature of the different components or limitations in the communication network, such redistribution may be hard or even infeasible.

The concept for scalable applications (see Figure 12) targets to exchange resources and QoS on a single node in the device-edge-cloud continuum, rather than redistributing applications differently across devices, edge, or cloud).

Resources are an abstract definition of different system properties such as:

- Amount of memory
- Amount and possibly types of computational resources
- Data bus bandwidth

QoS is abstract definition of different properties of the system output such as:

- Accuracy of the data
- Availability of system
- Latency of the system response
- Throughput of the tasks to be performed

Depending on which of these qualities need to be exchanged with a resource budget, different mechanisms need to be implemented. For example, a 5-tab two-dimensional filter (Two-dimensional filter, 2021) for image scaling could be replaced by 2-tab one-dimensional interpolation filter reducing the accuracy of the output pixel values. For this example, a reduced image quality is exchanged for a reduction in computational resources and memory bandwidth.

Another example is the algorithm for data compression. The highest compression ratio is achieved when the encoder can accurately predict data sample from past and future samples so that only small differences between the prediction and the real value needs to be transmitted. A better prediction can be achieved when more historic and future data is used in combination with complex prediction algorithms. Hence the

compression ratio can be exchanged with computation resources and latency that is introduced by using future data.

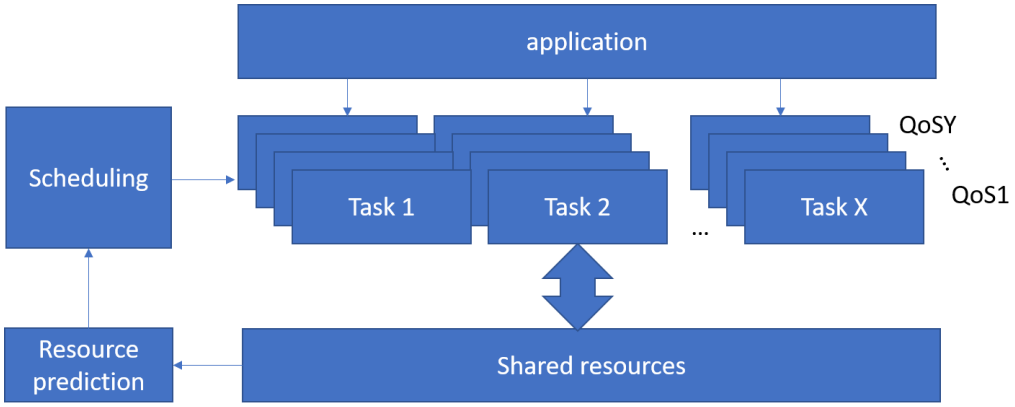

Figure 12. Diagram of the concept for scalable applications.

To introduce a scalable application in conjunction with the cloud, the following requirements should be satisfied (see also (Ferretti, Ghini, Panzieri, Pellegrini, & Turrini, 2010)):

- Different versions of a processing task are available, each with its only resource-QoS point
- The resource allocation is measurable and predictable
- A scheduling algorithm is available to select the needed resource-quality point, guaranteeing real-time performance.

In other words, scalable applications need run-time performance management (see section 6.4.3) to materialise the benefit of scaling. Note that the scheduling algorithm becomes more complex for systems with concurrent tasks as the scheduling will also involve load balancing.

Fit with concept TRANSACT reference architecture/components

In Figure 13, the orange ellipses show the components in three tiers of the TRANSACT reference architecture where the scheduling algorithm for scalable applications takes place. The processing tasks itself are the Functions in the upper layers of the tiers.

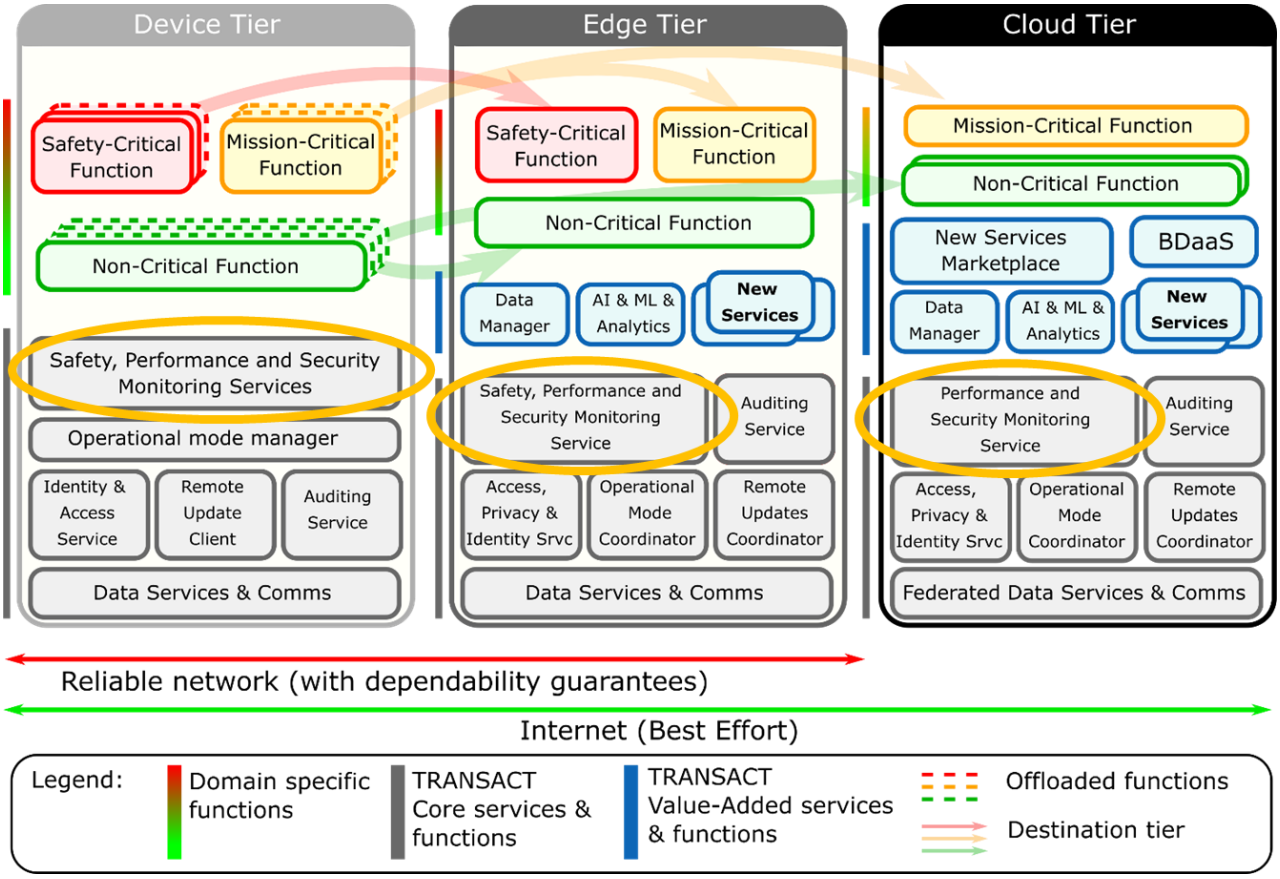

Figure 13. Scheduling of scalable applications in the TRANSACT reference architecture.

Example in context of a use case

Consider UC1 – remote operation of autonomous vehicles, the real-time behaviour and low latency for object recognition and navigation is extremely important for the safety. If the load of the processing units that process the sensor data is too high, an object might be detected too late or even missed and could cause a serious incident with even fatalities. The scheduling algorithm could decide to reduce the sampling rate of the sensor or reduce the resolution of the data to reduce the resource allocation. As a result, the quality is reduced as the maximum allowed speed of the vehicles is reduced.

Challenge for application within TRANSACT context

Typically, the performance of a device is based on overall system requirements. Dynamically changing the performance to limit the resource allocation might impact the overall system. Considering the above example, a reduced maximum allowed speed requires communication to the other parts of the system that actually make sure the vehicle does not exceed the maximum allowed speed. This considerably complicates the system.

5.7 Concepts for ensuring data integrity (across the device-edge-cloud continuum)

Overview

Artificial intelligence (AI) applications, and especially machine learning (ML) applications, are highly dependent on high quality data. In a typical ML workflow, data is collected and pre-processed for the purpose of model training. After the model has been trained and deployed, its prediction performance is only as good as the data it has been trained on. Therefore, ensuring and maintaining the integrity of data is critical for the success of ML applications.

There are different aspects of data integrity, and many potential reasons why data integrity may be compromised.

- Model training often uses heterogeneous datasets, for example, historical (batch) data, which is usually pre-processed, and streaming raw samples collected in (near) real time. These datasets may have different formats and data schemas, and different quality. This is often the case in distributed device-edge-cloud systems, where data originates from multiple sources and needs to be combined to be used. It is of importance that the data is consistent, and that the quality of the data is understood for the combination process. For example: in a hospital situation, in an operation, live data of a patient is combined with pre-operative (analysed/processed) data to help the surgeon do the procedure. Then it must be ensured that the pre-operative data is from the same patient as is being operated on.
- Data often contains missing values or out-of-range values, which needs to be identified and addressed before data can be used.
- Data may be corrupted, as the result of malicious attacks that manipulate data and make it false and unusable for the target purpose. For example, false data injection attacks (FDIA) are one type of such attacks which are hard to detect because they change the semantics of a message and preserve its syntactical correctness.
- In some applications such as healthcare, there must be a consent obtained for the purpose of data processing, which may be lacking.
- Data may be biased. While there are multiple definitions of data bias, it typically denotes a type of error such that some elements of a dataset are more heavily weighted than others, which negatively affects ML.

There are two essential steps to ensuring data integrity for the purpose of ML over its entire lifecycle, illustrated in Figure 14:

- Data quality monitoring;
- Data quality improvement.

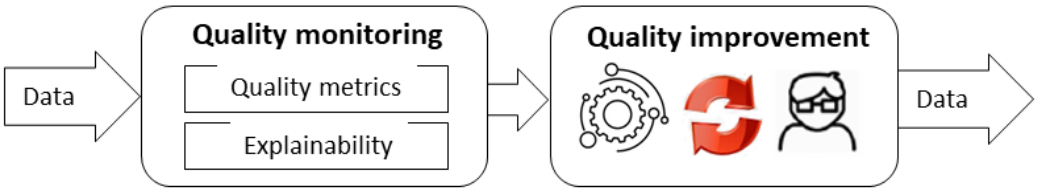

Figure 14. Ensuring data integrity for ML tasks

5.7.1 Data quality monitoring

One way to effectively identify data integrity issues is to implement data quality monitoring, for example to detect the mean shift in multi-attribute data (Yu, Wu, & Tsung, 2019) or to implement change-point detection for real-time monitoring high-dimensional streaming data (Zhang & Mei, 2020). Data quality monitoring needs to run both before and after ML model training and deployment in production. The metrics to be

monitored depend on the type of the data and the target domain. Examples of such metrics in the maritime domain, in the context of UC2, are the *frequency of Automatic Identification System (AIS) data* and *data noise.*

Frequency of AIS data: AIS enables tracking of vessels in maritime navigation, thus improving navigational safety. AIS data is collected via base station receivers (on land) and via satellite receivers. Monitoring this metric can help identify scenarios where no AIS data has been received for a given period of time, within a longer interval in which valid AIS data were recorded. No data received can mean that an AIS transponder is defective or turned off.

Data noise: AIS data can be duplicated and it can contain noise such as incorrect timestamp, speed over ground, position readings. This is sometime the results of dense traffic causing AIS message collision.

As part of data quality monitoring, one can provide explanations for the causes of data quality gaps, in order to help improve data quality.

5.7.2 Data quality improvement

Depending on the type of data quality issue, data improvement may be automatic or may require expert input. For example, if we are dealing with missing data and out-of-range data, this can be done automatically by predicting missing values and discarding out-of-range values. If the data monitoring indicates that a new class label is needed, in that case the expert can confirm adding a new class label for a given data point.

Fit with concept TRANSACT reference architecture/components

In Figure 15, the concept is positioned in the TRANSACT reference architecture in the Edge and Cloud Tier, connecting to data analytics components, AI&ML, and new services supported by data. Specifically, data quality aspects are of a high concern for data analytics solutions. High quality data insights rest on high quality data. Similarly, data quality aspects directly relate to AI/ML solutions, being one of their integral components. Finally, the potential of advancing business operations with new data services relies on high quality data. The purple rectangles in Figure 15 indicate the fit of the concept within the TRANSACT reference architecture.

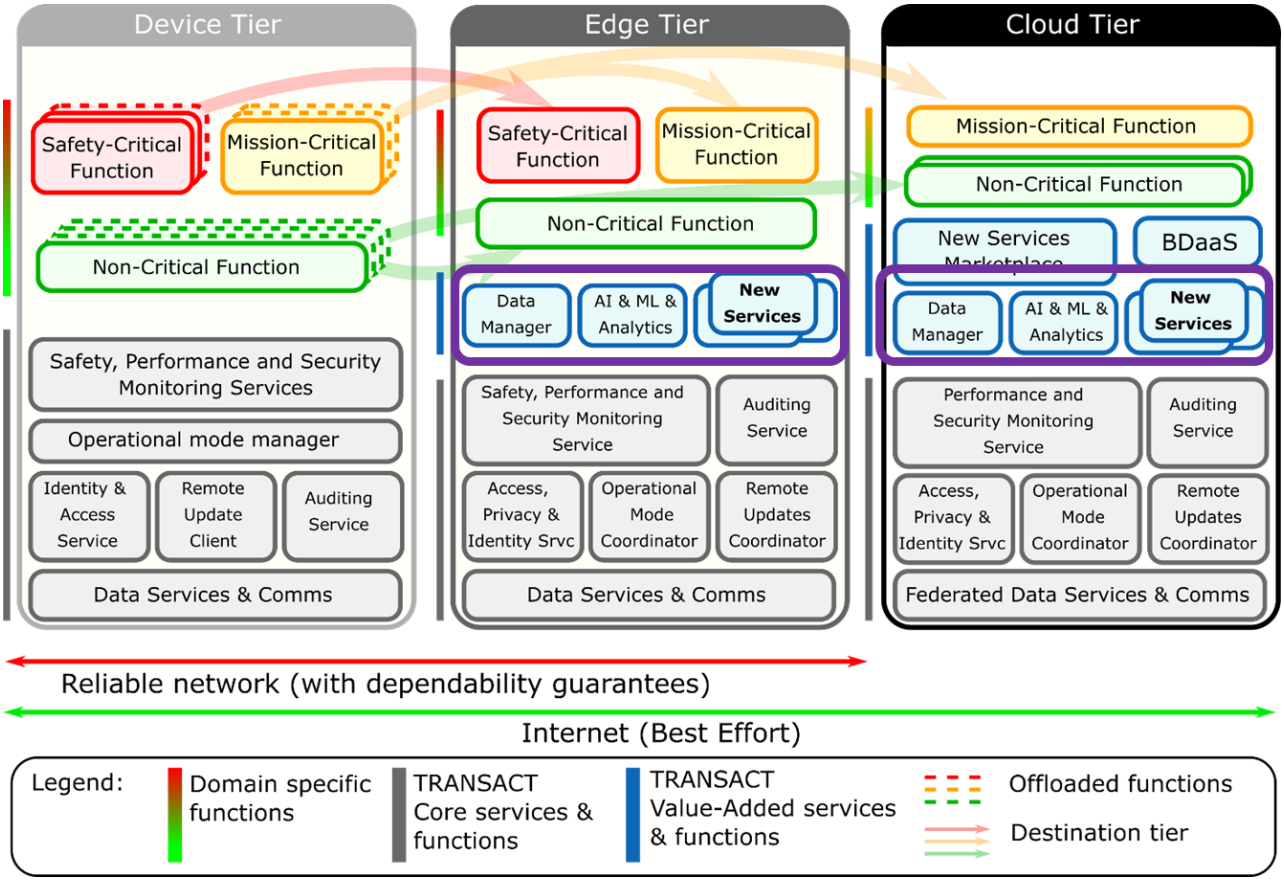

Figure 15. Data integrity in the TRANSACT reference architecture.

Example in context of a use case

We describe the concept using the maritime use case (UC2). UC2 deals with developing reliable routing advisory services that can help a ship navigator choose optimal shipping routes (based on criteria such as time saving, fuel saving, avoiding congested areas, avoiding bad weather, etc), while increasing safety at sea. ML-assisted models can greatly enhance the accuracy and efficiency of such service provision. Data quality monitoring and improvement plays an important role in developing these ML models, illustrated in Figure 16 and described below.

Developing ML-assisted routing advisory services requires a high volume of high-quality AIS data. This data is coming from different sources: historical AIS data and near real-time streaming AIS data. Each dataset needs to go through data monitoring and quality improvement before training of the target ML models, measuring a set of metrics. After the models are deployed in production, streaming data keeps feeding into the ML pipeline, going through data quality monitoring and data quality improvement processes.

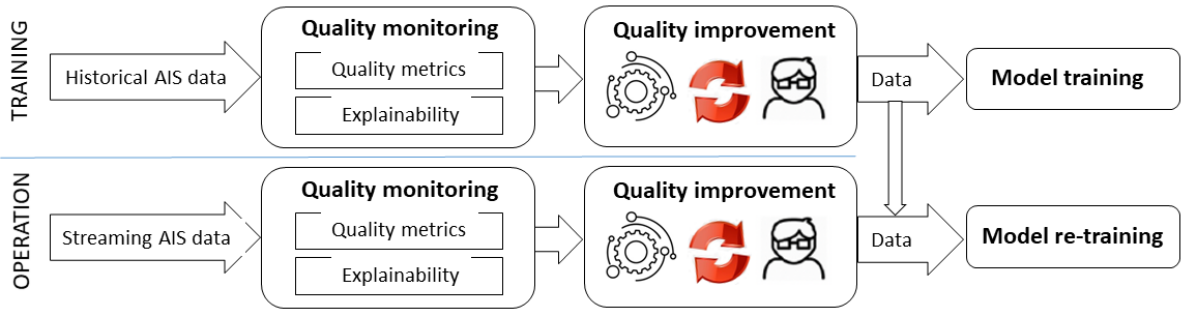

Figure 16: Data integrity for ML in maritime UCS2

Challenge for application within TRANSACT context

Some open challenges for the application of the concept for ensuring data integrity through data quality monitoring include selecting the metrics to be monitored, such to provide a useful monitoring service while not overloading the monitoring with too many details. Another challenge relates the question of how to provide human-centred explanations of data quality issues, which can be useful to understanding the reasons for data poor quality.

5.8 Concepts for AI monitoring

Overview

The typical AI lifecycle consists of many different steps, including data preparation, modelling and operations (see Figure 17) (Arnold, et al., 2020). Assessing AI model performance prior to deployment on unseen data (i.e. test cohort) is common practice, but monitoring performance post-deployment gets less attention. However, adequate monitoring in "real world" use is crucial to detect issues early and to retrain models at the appropriate moment, especially in safety-critical systems where inaccurate output may result in serious harm. There are several reasons why model performance may vary in various contexts or may degrade over time:

- **Limitations of training data:** In most applications, the collected training data only represents a portion of the real-world data to which the model will be exposed after deployment. Furthermore, data may have been carefully selected and curated for training purposes.
- **Model degradation:** The AI model performance often degrades over time. This can be due to data or concept drift.
    - **Data drift** can be defined as a variation in the input data during production as compared to the data that was used to train and validate the model. There are many factors that can cause data drift, but often it comes down to time. Training data will always be "past" data that may not be representative anymore for "current" data. For example, acquisition devices such as cameras or medical scanners may have evolved.
    - **Concept drift** happens when the relationships between the input and output data changes over time, resulting in less accurate predictions. This is for example the case when there is seasonal variation.

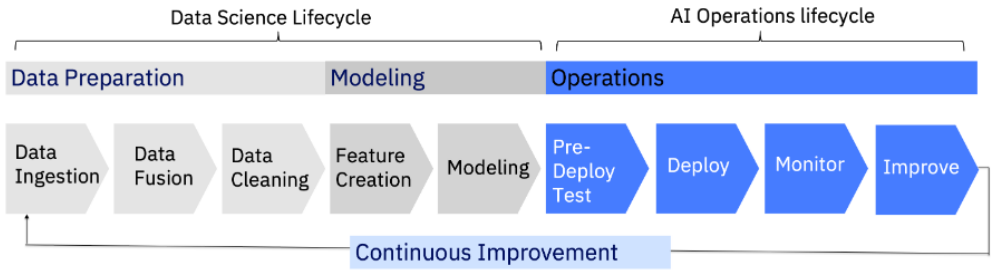

Figure 17. Diagram showing the typical AI lifecycle (Arnold, et al., 2020)

Performance criteria are required to determine whether the output provided by a deployed AI model is still acceptable. These performance criteria can originate from predefined performance requirements, but also other points of comparison can be used like performance on the test cohort or performance of a previous AI model version. Not meeting the performance criteria overall or in specific contexts or subgroups should trigger an alert, so that appropriate actions can be taken (e.g. re-training a model). The interaction between AI monitoring and performance criteria is visualized in Figure 18.

There are many metrics available for assessing the performance of AI models, such as precision, recall, F1 score, accuracy and area-under-the-curve (AUC). In applications with a human-in-the-loop where the AI output is reviewed by a person and corrected when needed, also the percentage of manual interactions could be used. Finally, it may also be important to look beyond the typical AI model quality metrics and include some higher-level business metrics or key performance indicators, such as customer satisfaction, as the AI output is often not directly exposed to the users but embedded in an application (Arnold, et al., 2020).

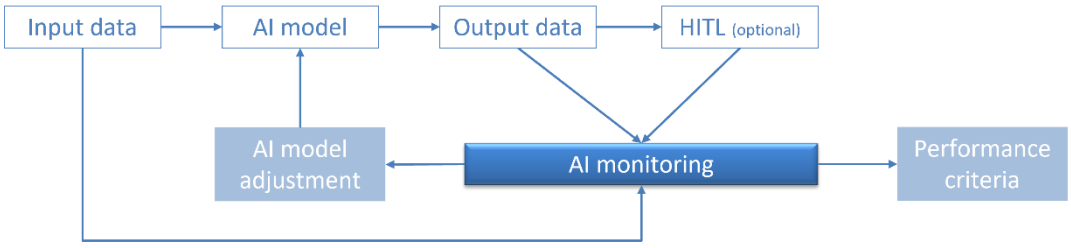

Figure 18. Conceptual diagram showing the central role of AI monitoring to drive AI model adjustments. (HITL = human-in-the-loop)

In order to compare actual model performance with the performance criteria, **ground truth data** is required. Getting such ground truth data can be challenging in some applications, while it can be continuously collected in other applications. Applications designed to speed up a manual process, reviewed and manually corrected when needed by a human-in-the-loop, continuously generate additional ground truth data during real-world use. In such applications, **continuous** monitoring can be rather easily implemented. For many other applications, monitoring AI model performance is a **periodic** process, that involves time-consuming manual labelling of data. In the absence of ground truth data, there are however alternative (indirect) ways to continuously monitor the AI output, for example by comparing the output with its expected statistical distribution (e.g. an output of 75 should trigger an alert if the typical output is 20±5).

In addition to monitoring the quality of the AI model output, it can also be useful to monitor the **input data**. By comparing the incoming data during real-world use with the data used during model training and validation, data drift can be detected on a continuous basis.

By using Gaussian analysis (Lee, Lee, Lee, & Shin, 2018) (van Amersfoort, Smith, Jesson, Key, & Gal, 2021) the AI model is also able to provide the user with a confidence score based on the Mahalanobis distance or uncertainty on the given predictions. This not only allows the detection of Out-Of-Distribution (OOD) samples (e.g. due to data drift), but also allows the human-in-the-loop to interpret how different the model perceives these samples when compared to the original training data.

Finally, also the concept of explainable AI (XAI) (Ribeiro, Singh, & Guestrin, 2016) (Samek, Wiegand, & Müller, 2017) is relevant in the context of AI monitoring. AI systems are often a black box to the end-user, and the purpose of explainable AI is to allow humans to better understand the model predictions. When changes in the performance are detected, the next step is identifying which features caused the drift. This analysis can be facilitated by using explainable AI systems as demonstrated in the reference above. The limitation of these systems is the often missing (yet required) pre-annotated text data, explaining what is wrong in ground truth data. An initial monitoring tool without explanation is therefore faster to set up, while ground truth data is being collected for the explained AI solution. In this TRANSACT project, an attempt on the initial monitoring tool is developed.

Fit with concept TRANSACT reference architecture/components

In Figure 19, the orange ellipses show the components in three tiers of the TRANSACT reference architecture where the AI monitoring takes place.

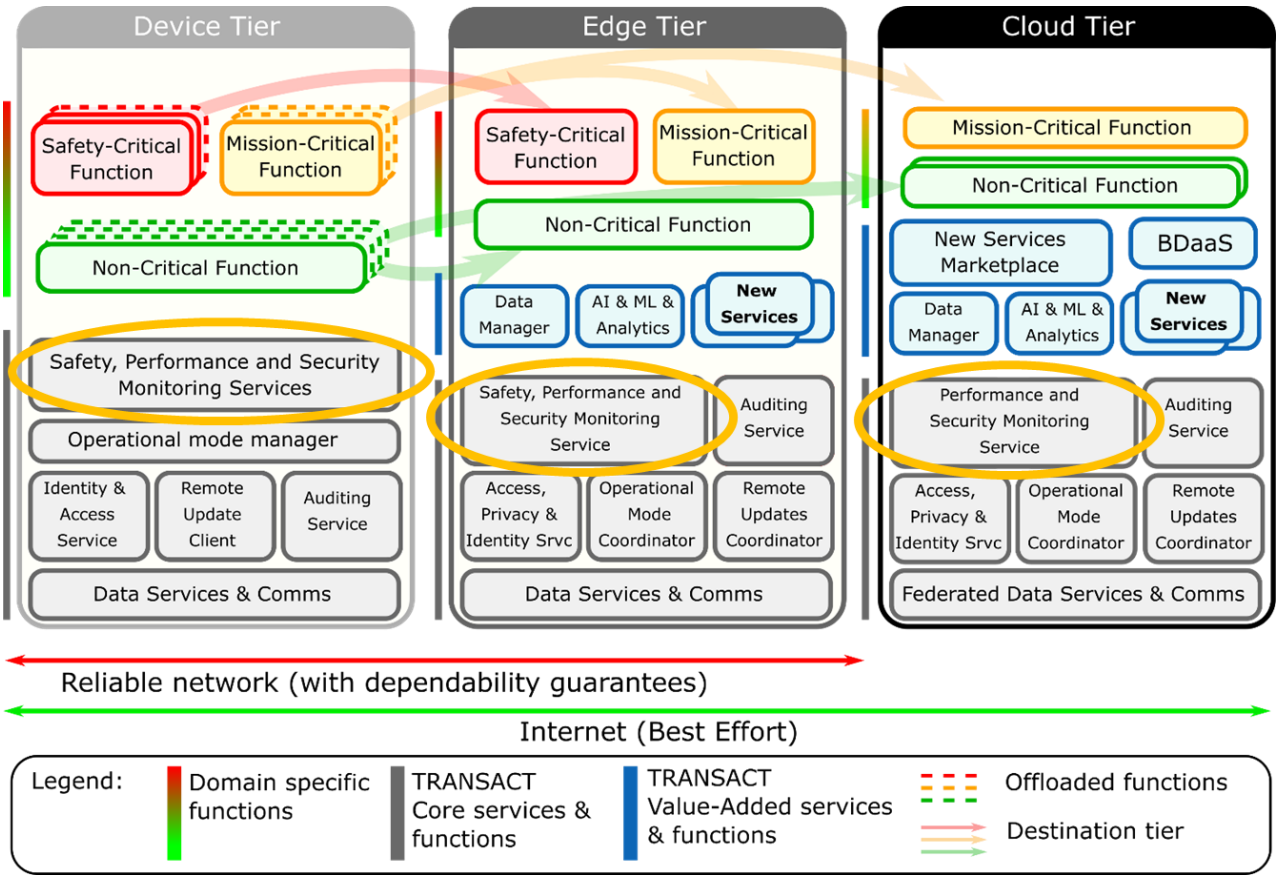

Figure 19. TRANSACT reference architecture.

Example in context of a use case

In this section, we will provide an example within the Medical Use Case (UC4). The pre-operative planning of structural heart interventions such as Transcatheter Aortic Valve Implantation (TAVI) requires medical imaging (e.g. CT) to analyse the anatomy of a specific patient and to take specific measurements. These measurements are required to determine the appropriate size of an implant (e.g. aortic valve annulus), and to assess the risks during the procedure (e.g. coronary heights). AI models can be very useful to increase the efficiency, by automatically performing an anatomical analysis including the relevant anatomical measurements. A potential workflow with a human-in-the-loop and continuous AI monitoring is shown in Figure 20, and described in the bullets below.

- An AI model detects a certain anatomical landmark on medical images
- The physician can inspect the AI output, as it is displayed in a cloud-based application together with the medical images, and correct it when needed
- The proportion of cases where corrections are needed can be continuously monitored, and also the magnitude of the correction can be monitored. This could then be compared with the performance criteria (e.g. for all cases, for a specific hospital, or for a specific physician) to detect concept drift.
- The input data (e.g. CT) distribution can also be monitored to measure data drift.

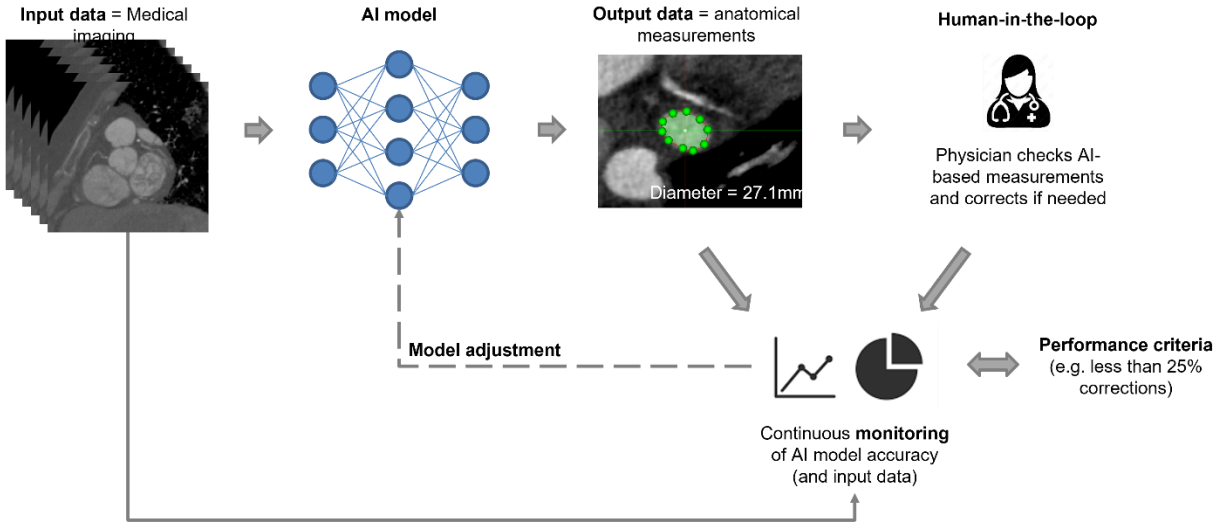

Figure 20: Example of AI monitoring in the context of UC4 (Medical use case).

Challenge for application within TRANSACT context

- How to react when acceptance (baseline) criteria are not met in certain situations?
- How to determine which subgroups / contexts to monitor, when a large amount of data has been generated, given that not everything can be monitored and detected (in other words, how to ensure to get relevant data back from the field)?
- How to use AI monitoring to quickly optimize performance in highly regulated environments?
- How to deal with large variety of incoming data (medical: resolutions, scanners, protocols)?

- How to generalize AI monitoring solutions across domains?
- How to handle the concept of model self-reinforcement (i.e. use of real-world unchanged AI data approved by human to re-train model)?
- How to prepare and evaluate robustness of newly trained models on possible data drift and concept drift?
- How to address some of the wider concerns for "AI in healthcare", such as data management, interoperability and trust and governance?

5.9 Concepts for safe and secure modular updates

Cyber Physical Systems (CPS) take over safety-relevant control tasks under conditions that change during the operating time (e.g. new traffic infrastructures for highly automated vehicles., new communication technologies, new sensor technologies or security mechanisms, extended application scenarios, etc.) Thus, the capability of updating these systems becomes necessary to keep such systems safe and up-to-date.

The ability to continuously incorporate experience from the operation of systems in the field into the further development of the CPS (swarm data collection and continuous updating,) and to design systems in such a way that improvements can be incorporated into the CPS frequently is particularly important. For the development of the updates, a process is necessary that considers the entire update process. The project Step-Up!CPS (Step-Up!CPS, 2021) proposes a process for safe and secure modular updates that covers all these aspects. The process is supported by methods and technologies that were developed during the project (Strathmann, et al., 2021).

Overview

In the following we give an overview of the process and shortly explain the concepts utilized in each of these steps (see Figure 21). The developed process is domain agnostic and aligned with the DevOps Cycle. When inventing this process relevant standards of the domains automotive, maritime and industry 4.0 as well as recommendations for Over the Air Updates were considered [(ISO, 2011) , (ISO, 2005) , (IEC, 1998) , (IEC, 2016)] .

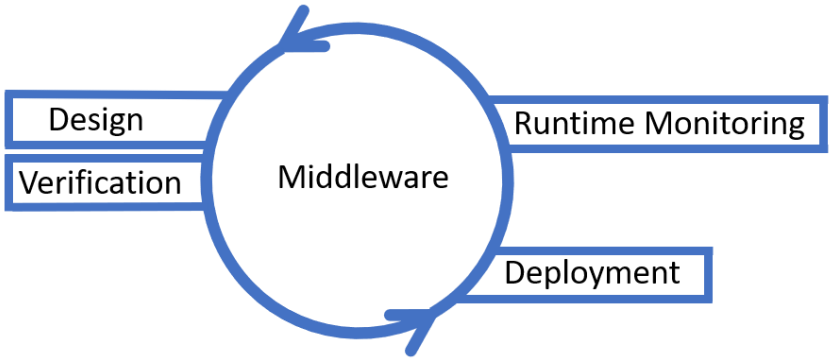

Figure 21. Process for Safe and Secure Updates

A major challenge with updates is to keep the re-verification effort low. It is not reasonable in terms of time and cost to carry out a complete reverification of the whole system. To tackle this challenge the process is designed to allow modular updates. The module is regarded as the smallest unit to be updated. The boundaries (the behaviour exhibited at the interfaces) of the module are formalized with suitable

specification methods, while the necessary re-verification activities are identified by means of performing an impact analysis to identify the necessary reverification activities. In other words, for a new functionality or improvements to existing functions only the new or improved functionality alongside all affected functions has to be considered for the re-verification which is necessary to meet the safety standards.

Middleware: The Step-Up!CPS Middleware plays a central role in the process. A wide variety of hardware platforms are used for CPS. It has to be made sure when developing and deploying updates that they are compatible with the underlying hardware. Hardware virtualization and services in a CPS middleware are used for the compatibility checks and deployment of the update. The middleware services allow to abstract from the underlying hardware and helps to match the needed and provided hardware resources. The middleware offers services to check during the deployment if the update is compatible with the target hardware.

Design: The design of the updated component is triggered either by errors in system behaviour or the request for new functionality. The design phase starts with describing the functional, timing and safety aspects of the updated component. The realization of components puts very specific requirements on the provided hardware resources. Contract based design (CBD) [(Meyer, 1986), (Benveniste, 2012)]is used to capture the specification of the components along the design perspectives starting from the functional and down to the technical perspective.

CPS interact with their environment and can cause severe damage. With the increasing level of automation of these systems, the paradigm of fail-safe mechanisms is no longer sufficient and fail-operational mechanisms becomes necessary. The Step-Up!CPS process allows to consider fail-operational mechanisms early at design time. These mechanisms pose very specific requirements on the underlying hardware like timing, resource demand and segregation properties. Contract-based design enables specifying these properties early in the design process when little knowledge about the underlying hardware is available. A central part of the used fail-operational concepts is runtime monitoring. Runtime monitors, which can be automatically generated from the component specification are used as a part of the safety concept for fail operational mechanisms like the activation of redundancy mechanisms or degraded operation modes. Monitors are also used for collecting live data from the field during runtime. The data can be utilized for identifying the need for future updates, which makes the monitors an integral part for improving the system.

To abstract from the concrete hardware the concept of services is used. The service concept allows to design the system in the absence of concrete knowledge about the hardware in early design phases. The needed hardware resources of the system can be specified before knowing the exact target hardware. On a technical perspective the required services are mapped to the provided services of the middleware stack which abstracts from the concrete hardware. The specification of these services is done via the contracts of the involved components.

Verification: During the verification of the updated systems, there is a necessity to check whether the specification is fulfilled or not. A Virtual Integration Test (VIT) plays a central role in that regard. A VIT checks whether the contracts of the system and its (sub)components are still consistent. The concept of Incremental Contract-based Verification is used to reduce the reverification effort by specifically targeting the integration impact of the introduced changes by checking their compliance with the contractually agreed assumptions and guarantees [(Bebawy, et al., 2020), (Guissouma, Kroger, Maelen, & Sax)]. The verification of the implemented system itself is done via simulation. Within the simulation the contracts can be used to constantly check if the implementation fulfils the specification.

Deployment: The update package is signed and transmitted to the target system via an authenticated and encrypted connection. Server and device certificates enable the identification, legitimacy of the parties involved and ensure that only the correct update package is installed. In addition, a Device Configuration

Identifier is used to determine that the update may only be updated on the designated variant of the client. The Step-Up!CPS middleware performs checks to assure the resource demand of the updated system matches the provided resources of the target hardware.

Runtime Monitoring: Runtime monitoring capabilities allow to detect violations of the system specification during runtime. Monitoring information can be used to trigger safety mechanisms. The gathered information is also used to identify the need for an update. Monitoring the updates system during runtime is an integral part of the lifecycle of the CPS. If conditions are detected during runtime which make an update necessary, it is tracked by the runtime monitors and reported to the developers through a feedback mechanism.

Fit with concept TRANSACT reference architecture/components

In Figure 22, the blue ellipses show the components in three tiers of the TRANSACT reference architecture where the Remote Update takes place. Apart from the application of the update which takes place at the marked components the update is as we described part of the whole design process.

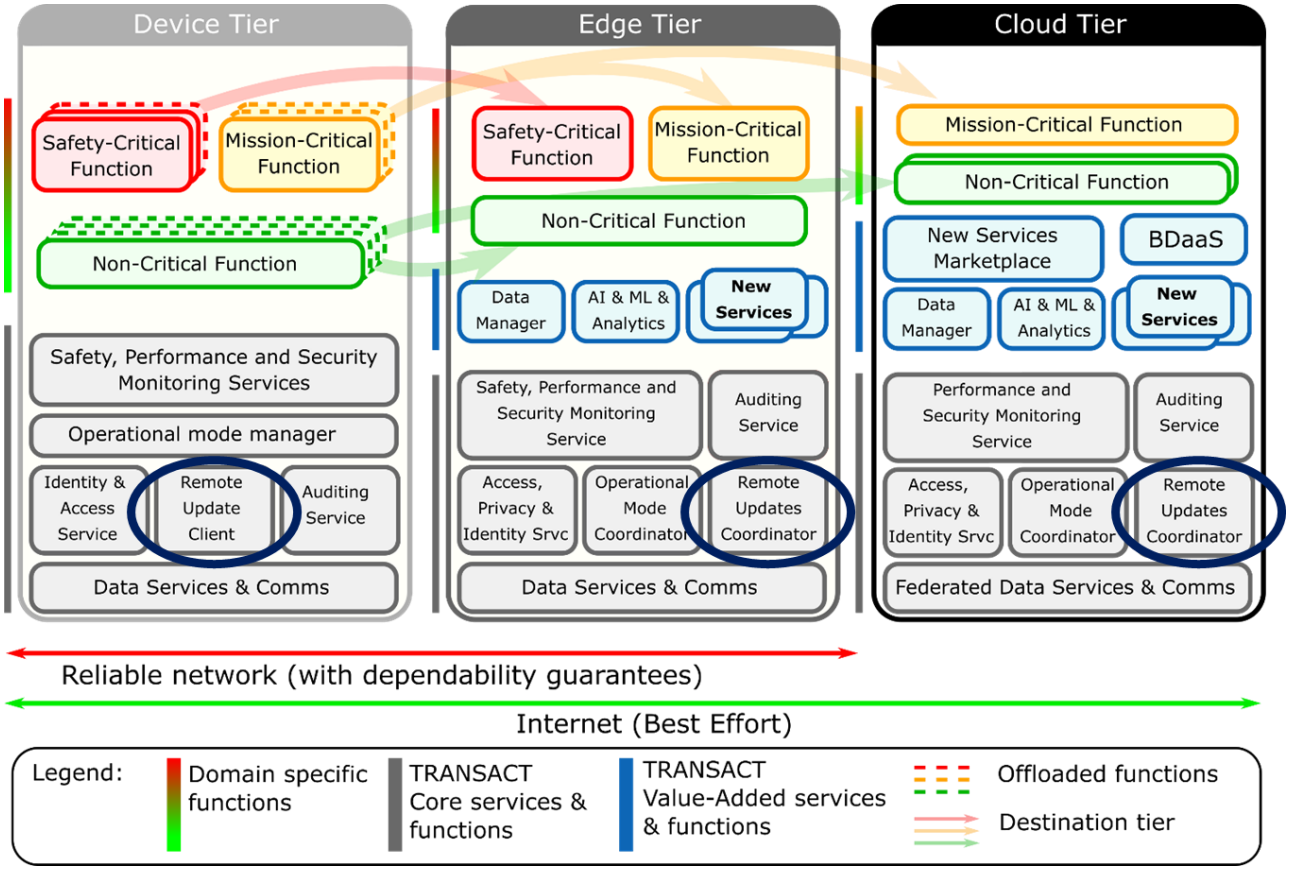

Figure 22. Positioning the safe and secure updates in the TRANSACT reference architecture.

Example in context of a use case

All Use Cases are planning to introduce new cloud-based functionality into their Use Case. The proposed methodology in this section covers the whole DevOps Cycle from development, implementation, verification, deployment to runtime monitoring in the field.

In the context of the TRANSACT project this can be exemplified well for Use Case 3. Use Case 3 (cloud-featured battery management system) aims at collecting data from a fleet of vehicles for analysing these

data for updating the software as well as improving the user knowledge about the health status of the battery management system itself. The knowledge gained from analysing the collected data can be used for the development of new updates which need to be rolled out in a safe and secure way. This is captured by the proposed process.

Challenge for application within TRANSACT context

The TRANSACT use cases are extending systems with cloud functionality which is often using machine learning and AI technologies. By this the systems can be extended with capabilities that allow to proceed to mission-oriented and cooperative systems. Mission-oriented systems in e.g. the automotive domain no longer perform only certain driving functions (e.g. lane keeping, ACC, emergency braking assistance, etc.), but navigate autonomously and self-responsibly through traffic. This requires a semantic understanding of the operational space, which is increasingly used and processed with the help of AI methods for e.g. the implementation of driving tasks.

The concepts developed so far for the specification and evaluation of system capabilities must be extended to meet the requirements for the application for mission-oriented or cooperative systems.

6 Cross-cutting concepts for safety and performance

6.1 Introduction

This section describes cross-cutting concepts.

Cross-cutting concepts are system-level methods and techniques for linking application and platform. They include concepts for designing and deploying the application on the platform, as well as analysing and run-time monitoring and managing the behaviour of the application running on a specific platform in the device, edge, cloud continuum.

6.2 TRANSACT project harmonised needs and expectations

As part of the TRANSACT project and following the TRANSACT reference architecture, the safe and correct deployment and operation of distributed applications on a distributed device-edge-cloud continuum must be ensured. For this relevant safety and performance concerns, both at design and at run-time, must be addressed.

Predictable end-to-end performance across the device-edge-cloud continuum is such a key concern. Dynamic end-to-end safety and performance monitoring at runtime linked to real-time performance management is needed. Complementary, safety assessment is needed to identify potential risks and hazards, and at run-time health and safety controls are to monitor these for the safety and health of the distributed solution, and support operational mode management, including fallback to degraded modes when necessary. Finally, a key aspect for achieving rapid innovation is that the necessary proof for safety assurance of incremental updates can be provided efficiently, i.e. to be suitable for low-effort, reduced cost, frequent re-qualification.

The complete set of technical requirements is documented in TRANSACT deliverable D1.2. This section describes selected cross-cutting concepts that address these requirements and lists open challenges for these concepts as applicable that will be investigated in the course of the TRANSACT project.

6.3 Selected cross-cutting concepts for safety and performance

The selected cross-cutting concepts are highlighted in Figure 23.

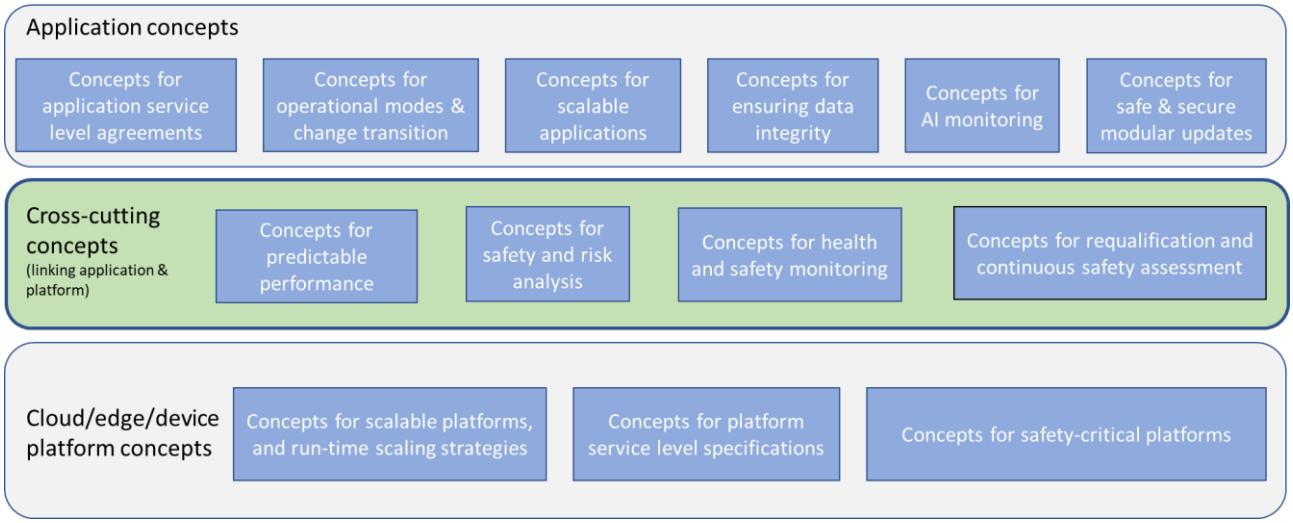

Figure 23: Selected cross-cutting concepts

6.4 Concepts for predictable performance

"System performance often brings the competitive advantage for high-tech Cyber-Physical Systems (CPS). To meet market demands for product quality, product customization, and total cost of ownership per product, systems need to meet ever more ambitious performance targets relating to system productivity. Performance is a cross-cutting system-level concern, with intricate relations to other system-level concerns like product quality, cost, reliability, security, and customizability" (Sanden, et al., 2021). Typical examples of performance aspects are end-to-end latency or response times, throughput, and scalability. Performance aspects may be of different levels of importance or criticality. For real-time systems, timing performance is an integral aspect of the functional behaviour of the system. For safety-critical systems, performance aspects may be integrally essential to safety of a system.

In distributed CPS solutions, designing a system for required or optimal performance is extremely challenging. Their distributed nature makes performance an emergent property of the interaction of many components that are physically distributed, often heterogeneous, and not necessarily subject to a single point of control. Some of the resources, such as cloud resources, are shared and managed in unpredictable ways and may not always provide the levels of reliability required by an application. The satisfaction of performance requirements and optimization of performance objectives can therefore not be dealt with at design-time only, but also requires active run-time management.

Model-driven system-performance engineering for Cyber-Physical Systems (MD-SysPE) is a methodology encompassing modelling formalisms, methods, techniques, and industrial practices to design for performance (Sanden, et al., 2021). Figure 24 depicts five key focus areas, relevant for MD-SysPE, in the V-model system development process (taken from (Sanden, et al., 2021)).

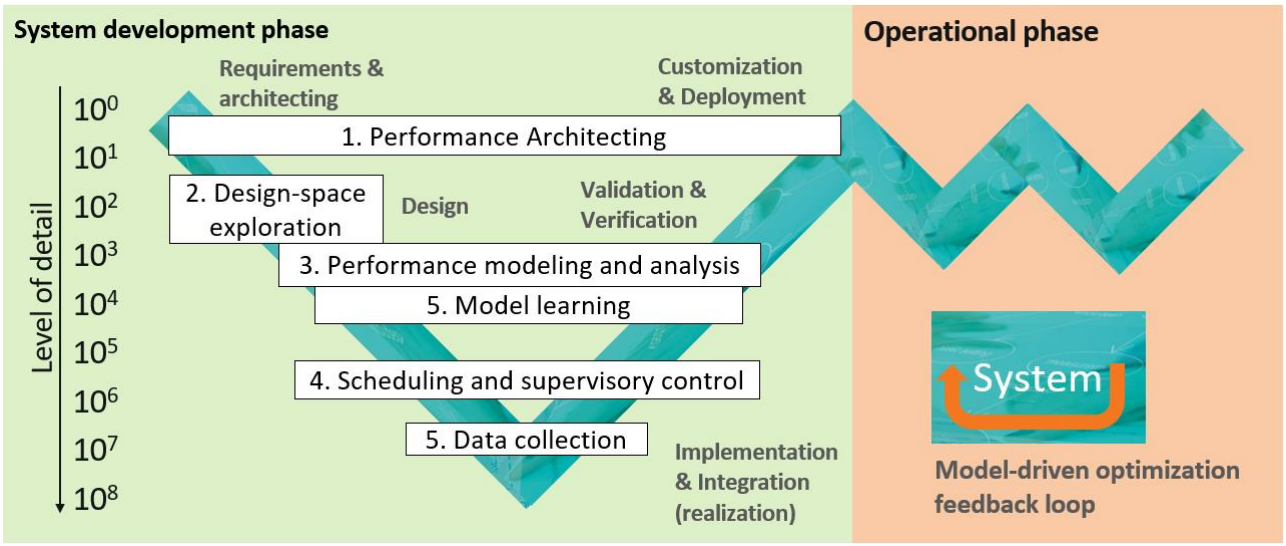

Figure 24: MD-SysPE focus areas in V-model (from (Sanden, et al., 2021))

In brief, these focus areas are

1. Performance architecting to determine the performance aspects that need to be taken into account at the start of the development process and during the system life cycle.
2. Design-space exploration to explore the trade-offs and find optimal designs within a given system architecture.

3. Performance modelling and analysis to express and analyse the performance of specific system configurations.
4. Scheduling and supervisory control *to achieve the required performance during system operation.*
5. Data-driven analysis and design to enable model learning, model validation and model calibration.

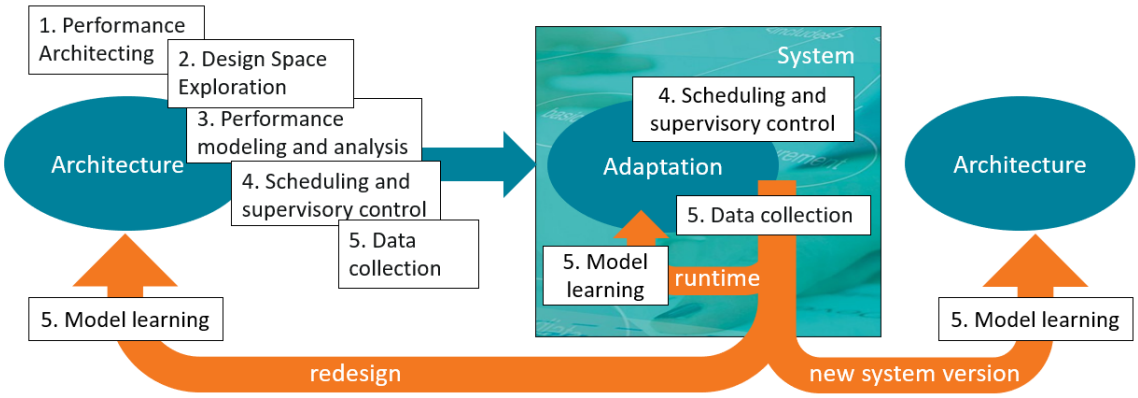

Figure 25: System life cycle with a positioning of the MD-SysPE focus areas (from (Sanden, et al., 2021))

Figure 25 positions these focus areas in the system life cycle, emphasizing the feedback cycles from system operation to improve system performance. Operational data may be used to improve system performance at runtime and through system updates. It may also serve as valuable input for the development of new systems.

In this section, we will highlight three concepts that are fit for the device-edge-cloud continuum:

- Concept for *performance monitoring* to actively monitor performance and detect degradation or violations. These concepts correspond to focus area 5 in Figure 24.
- Concepts for *performance modelling and prediction* to predict relevant aspects of performance of an application or platform, dependent on alternative design-time or run-time situations. These concepts correspond to focus area 3 in Figure 24.
- Concepts for *performance management* to influence the relevant aspects of performance of an application to adjust to run-time situations, and to stay within limits of performance requirements. These concepts correspond to focus area 4 in Figure 24.

6.4.1 Performance monitoring

Overview

The TRANSACT approach to the development of distributed CPS includes *performance monitoring* as a solution towards the challenges of meeting performance requirements and objectives. It corresponds to focus area 5 of MD-SysPE depicted in Figure 24. *The goal of performance monitoring is to observe the temporal behaviour of a system, taking advantage of the statistics provided by performance monitors* (Valente, et al., 2021). These monitors are watchpoints that collect system metrics or events, typically characterized by the time they occurred, metric values the type of event, and any additional attributes required to describe it. Tools that support performance monitoring are, e.g., OpenTelemetry and Prometheus (Chakraborty & Kundan, 2021).

Performance monitoring is an important part of the well-established MAKE-K cycle (Figure 26). This cycle defines the main four phases that an adaptive system performs: *Monitor-Analyse-Plan-Execute* (Kephart & Chess, 2003). The cross-cutting concept that is shared by these phases concerns the *Knowledge* that the system has about itself and its context.

Consistent with the TRANSACT reference architecture, *performance monitoring* in TRANSACT is concerned with the *Monitor* and *Analyse* phases. The *Plan* and *Execute* phases together are called *Performance management* in TRANSACT which is the topic of Section 6.4.3. Since both the *Analyse* and *Plan* phases are concerned with performance modelling and prediction, we dedicate Section 6.4.2 to this important topic.

This section focuses on the *Monitor* phase, that defines the observation of the system itself and its environment. The *Analyse* phase during which the collected data is processed is briefly discussed as well but is further elaborated in Section 6.4.2.

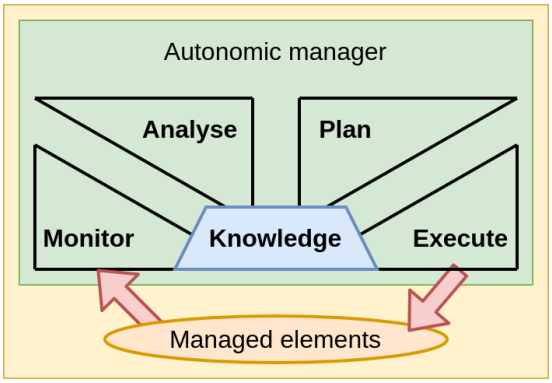

Figure 26: MAPE-K autonomic loop (from (Kephart & Chess, 2003))

Essentially, the monitoring process consists of the following steps (adapted from (Kornaros & Pnevmatikatos, 2013)) depicted in Figure 27:

- Capture an event by a periodic monitor or an asynchronous event trigger. Different types of monitors distributed across the system are in charge of collecting raw information from the system itself and its environment.
- Represent the event in a defined format. To process the data collected from different sources and methods, the data must be homogenized.
- Filter or pre-process the captured data. The essential information is filtered of the complete set of data.
- Pre-processed data can either be stored or transmitted to be used by other components.

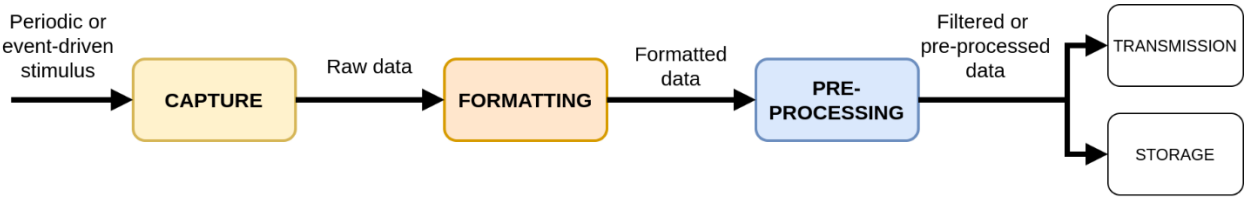

Figure 27: Run-time monitoring process (adapted from (Kornaros & Pnevmatikatos, 2013))

Different parameters characterizing the monitoring process are relevant to consider (Falcone, Krstić, Reger, & Traytel, 2021). These include:

- Type of instrumentation: hardware, software or hybrid monitors. Software monitors are usually more intrusive than hardware monitors, depending on the coupling between the monitor and the monitored system. On the other hand, full hardware monitors do not support the captured data to be easily used at the application level.
- Monitoring architecture: centralized or distributed. Specifying the architecture in terms of distributed interdependent modules instead of a monolithic structure increases flexibility, but also the complexity of the system.
- Synchronization: local or global. Important topics concern the use of clocks, their synchronisation and methods to compare timestamps generated from different sources.
- Publishing format: raw traces or pre-processed statistics. Depending on the volume of the dataset, the number of statistics may have to be reduced before the analysis phases.

Regarding the *Analyse* phase, important topics in current research literature are *run-time verification* and *self-aware systems*. The first, also known as run-time monitoring, is the study of methods to analyse the dynamic behaviour of computational systems (Falcone, Krstić, Reger, & Traytel, 2021). Dynamic behaviour includes temporal behaviour, which is related to the fulfilment of temporal constraints defined by non-functional requirements. The latter describes systems with the ability to adapt (in the Plan and Execute Phases) to different operating environments autonomously, based on the characteristic of self-awareness (Bellman, et al., 2020). To do so, concepts such as self-monitoring or self-modelling are developed, that allow the observation of the real behaviour of the system and to compare it with the expected one.

Fit with concept TRANSACT reference architecture/components

In Figure 28, the purple ellipses show the components in three tiers of the TRANSACT reference architecture that are responsible for performance monitoring. The *Safety, Performance and Security Monitoring Service* components capture, format and pre-process data collected from the system, i.e., carry out the Monitor phase in Figure 26. These components can include the *Analyse* phase in Figure 26 to analyse/predict whether performance properties are met. In case of a (predicted, future) violation of a performance requirement, the component will signal an Operational Mode Manager/Coordinator component that takes action to prevent future violations), carrying out phases *Plan* and *Execute* in Figure 26.

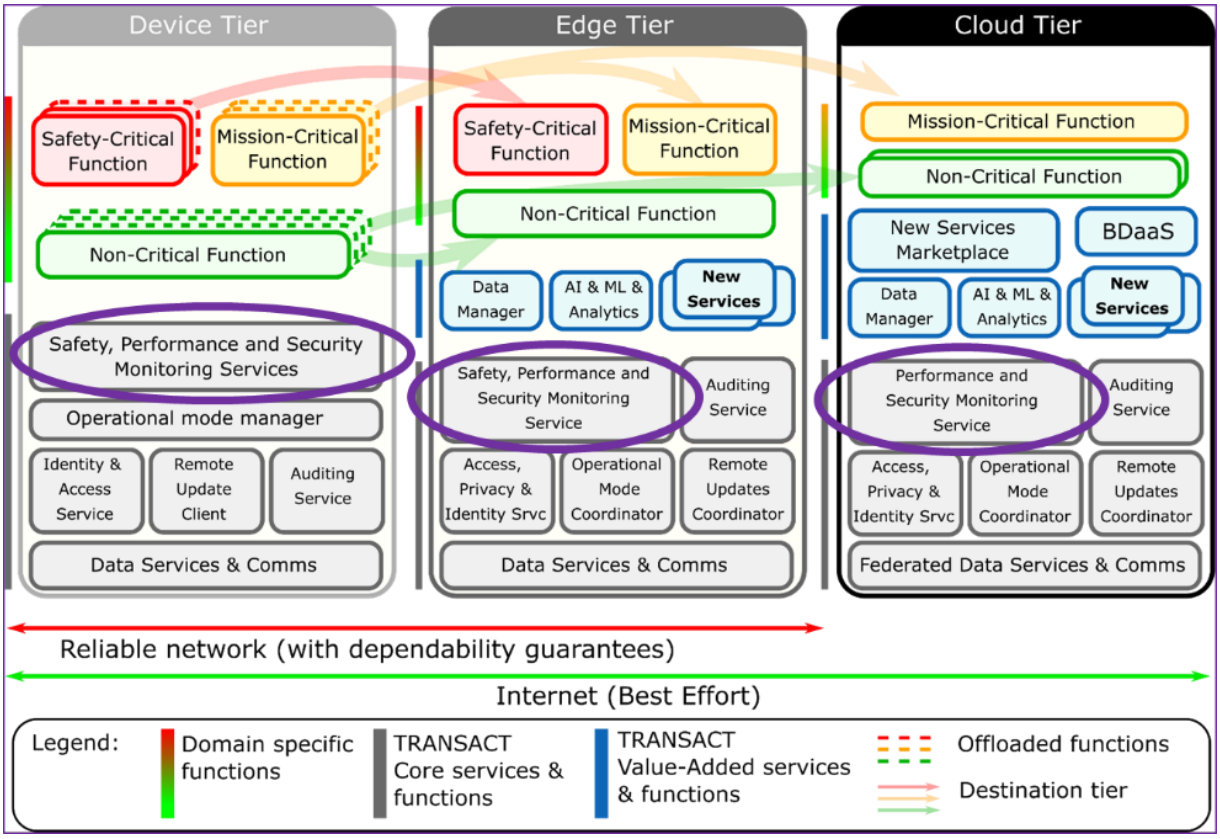

Figure 28: Performance monitoring in TRANSACT reference architecture

Example in context of a use case

The need for performance monitoring can be illustrated by the use case scenario 'Automatic image registration' of UC4. This scenario is concerned with the ability to integrate pre-interventional 3D CT images with the live X-ray image guidance so that a cardiologist can perform the required treatment on the patient. Upon request, the system must be able to perform the image registration in the cloud, with a response time of at most 5 seconds. As a fall-back scenario of automated registration, the cardiologist should be able to perform a manual registration.

Performance monitoring has an important role in this use case. The *Safety, Performance and Security Monitoring Service* components monitor the response time of the image registration and could perform run-time verification of the timing requirement of 5 seconds, and (self-)modelling of the response-time statistics, taking into account response time variations. In case the probability that the response time exceeds 5 seconds is predicted to be too high, the component can trigger an Operational mode manager, which can adjust the allocation of platform resources, to keep the response time requirement in check.

Challenge for application within TRANSACT context

Performance monitoring in TRANSACT requires

- Determining the intrusiveness of performance monitoring, by getting insight to what extent the (timing) behaviour of the system is affected.
- Time synchronization mechanisms to allow time-synchronized data collection in the distributed CPS.

- Efficient performance model learning techniques, that can be encapsulated in run-time monitors.
- Efficient performance prediction techniques, that can be encapsulated in run-time monitors.

6.4.2 Performance modelling and prediction

Overview

The TRANSACT approach to the development of distributed CPS includes *performance modelling and prediction* as a solution to the challenges of performance requirements and objectives. *The goal of performance modelling and prediction is to predict performance qualities of a system, dependent on system settings such as resource allocation, quality settings of an application or operational modes.* Modelling and predicting performance is a key area in MD-SysPE (corresponding to focus area 3 in Figure 24), but is also an important ingredient in performance monitoring (see Section 6.4.1) and run-time performance management (see Section 6.4.3) and corresponds to the *Analyse* and *Plan* phases in the MAPE-K cycle depicted in Figure 26.

Performance modelling and prediction for distributed CPS is facilitated by the *Y-chart paradigm* depicted in Figure 29. The Y-chart paradigm (Hendriks, Basten, Verriet, Brassé, & Somers, 2016) (Kienhuis, Deprettere, Vissers, & Wolf, 1997) (Lapalme, et al., 2009) proposes to model application functionality and the implementation platform as separate elements, with an explicit mapping as variation point between them. This allows easy variation of application functionality, platform resources, and mapping choices and facilitates analysing the performance impact of these choices, forming a convenient basis for (automated) design-space exploration to systematically explore design alternatives around these variation points.

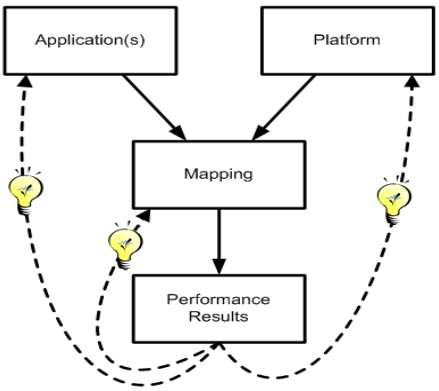

Figure 29: Y-chart based performance modelling

The structure imposed by the Y-chart can be combined with numerous performance modelling approaches such as data flow, timed automata, stochastic processes, queuing networks, discrete-event simulation and machine learning approaches. These different approaches can model different CPS characteristics, support different properties to be analysed, and with different degrees of accuracy and efficiency.

Fit with concept TRANSACT reference architecture/components

Figure 30 shows the components in the three tiers of the TRANSACT reference architecture that benefit from performance modelling and prediction. The *Safety, Performance and Security Monitoring Service* components may encapsulate (learned) performance models, enabling the prediction of whether certain performance properties (concerning, e.g., response latencies or throughput) will be met. In case of a (predicted, future) violation of a performance requirement, an *Operational mode manager/Coordinator component* can decide to transit between operational modes. The optimal configuration parameter settings (i.e., optimal mode) can be determined by an encapsulated performance model that allows the performance

to be predicted for different configurations. Notice that performance modelling and prediction apply to the complete TRANSACT reference architecture as well, e.g., to dimension the three-tier execution platform and to design-time deploy the safety-critical CPS application. This overall aspect is not shown in the figure.

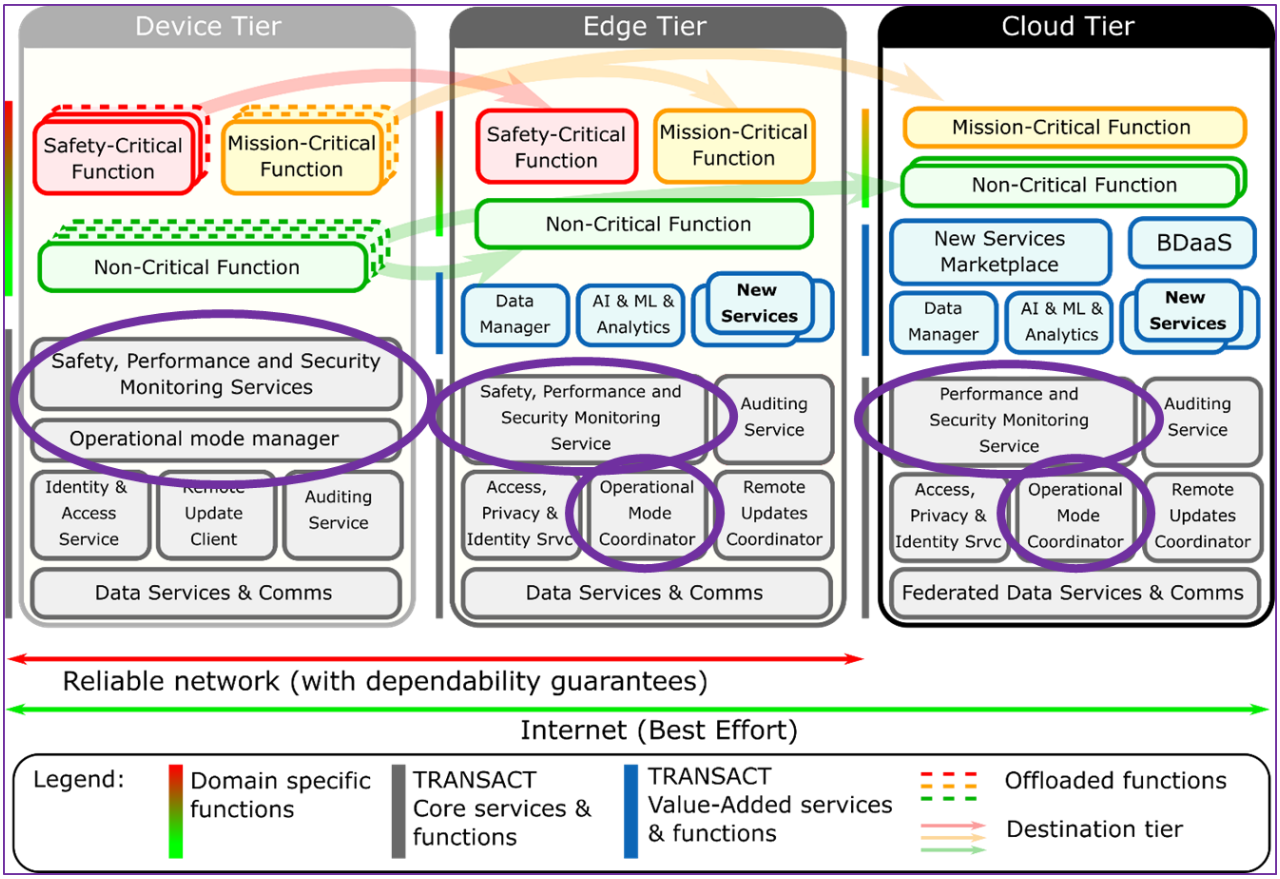

Figure 30: Performance modelling and prediction in TRANSACT reference architecture

Example in context of a use case

The need for performance modelling and analysis can be illustrated by the use case scenario 'Image reconstruction for treatment and diagnosis' of UC4. This scenario is concerned with delivering a 3D reconstructed CBCT image upon request from a radiologist / cardiologist within a reasonable timeframe (typically 20 seconds after acquisition). The 3D reconstruction application consists of several components and services to be allocated on resources over the edge/cloud continuum.

Determining an optimal deployment of application components on the platform and assigning appropriate resource shares in such a way that response-time requirements are met is challenging. Predictive performance models are instruments to underpin the decisions and explore the vast number of alternative choices in a systematic manner.

Challenge for application within TRANSACT context

Performance modelling and prediction in TRANSACT requires

- Performance models capable of predicting relevant performance qualities with sufficient accuracy, dependent on system settings such as resource allocation, quality settings of an application, or operational modes.

- Performance models capturing the variation in performance qualities, which are intrinsic to the edge/cloud- based systems.
- Systematic ways to obtain performance models, by first-principle modelling, by model inference using data measurements or by a combination thereof.
- Off-line model-based performance prediction techniques for design-time optimization, allowing relevant performance qualities to be predicted with sufficient accuracy, dependent on system settings.
- On-line model-based performance prediction techniques for efficient runtime performance monitoring and management, allowing relevant performance qualities to be predicted efficiently and with sufficient accuracy, dependent on system settings.

6.4.3 Performance management

Overview

The TRANSACT approach to the development of distributed CPS includes *performance management* as a solution to the challenges of performance requirements and objectives. Performance management is part of focus area 4 of MD-SysPE depicted in Figure 24 and is concerned with the *Plan* and *Execute* phases of the MAPE-K cycle depicted in Figure 26. *The goal of performance management is to influence the relevant aspects of performance of an application to adjust to run-time situations and to stay within limits of performance requirements.* Management strategies may include adjusting resource allocation or quality settings of an application, shaping, or balancing of workload. Performance management may involve continuous adjustment of system settings, but also reconfiguration of operational modes of the system or application, for instance placing functions at different tiers, such as edge or cloud, in which case the reconfiguration process itself may be subject to performance constraints.

Performance management can be explained generically though the Quality and Resource Management (QRM) architecture shown in Figure 31. This reference architecture is developed in the FitOptiVis (ECSEL Joint Undertaking, 2018-2021), and follows the Y-chart decomposition as depicted in Figure 29. Application and platform components have explicit QRM interfaces. A QRM interface has six aspects: inputs, outputs, provided budgets, required budgets, qualities, and parameters. Parameters capture the configurable setpoints (i.e. modes) of a component. Qualities capture all component properties of interest, like performance, safety, image quality, cost, etc. Input and output capture the functional aspects that are of interest for QRM, like data resolutions and rates. Provided and required budgets capture resource (e.g., processing, bandwidth, storage) provisions and/or needs. Each configuration of a component specifies a relation between those six aspects, i.e. can be taken to represent a particular Quality-of-Service level for an scalable application (see Section 5.6). The reference architecture is supported by a modelling language (QRML – for QRM Language) and toolset ((Berg, et al., 2020), https://qrml.org) that allows to model the QRM view of a system. QRML has a well-defined mathematical semantics framework (Hendriks, Geilen, Goossens, de Jong, & Basten, 2021) that supports multi-objective optimization. The framework is component-based, supporting both design-time and run-time reasoning and optimization.

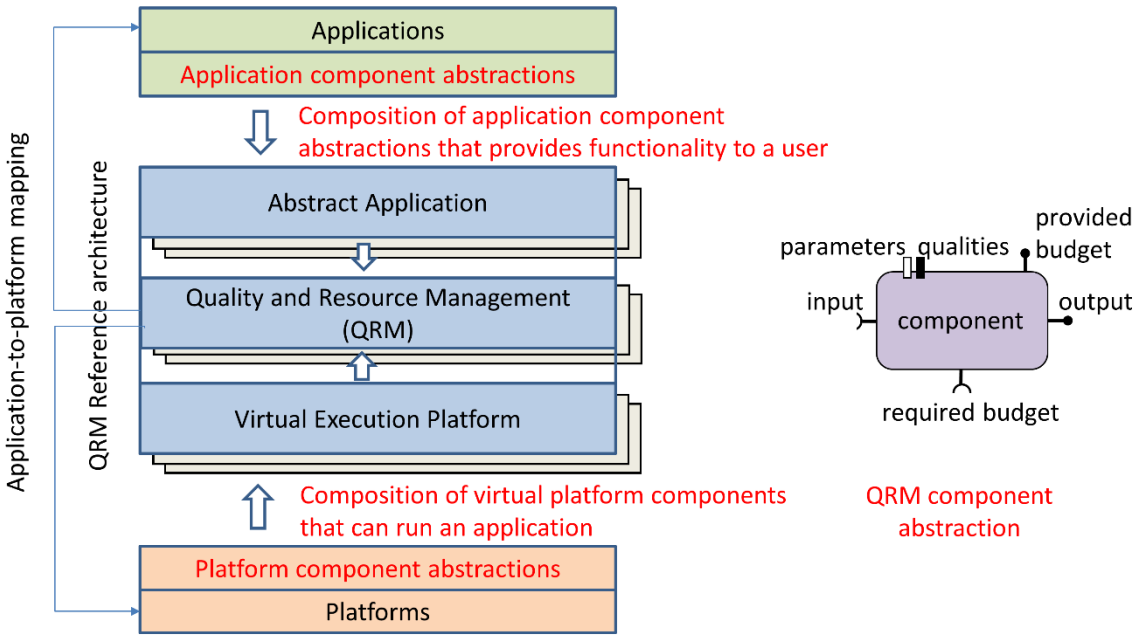

Figure 31: QRM reference architecture (from (Sau, et al., 2021))

Fit with concept TRANSACT reference architecture/components

In Figure 32, the purple ellipses show the components in three tiers of the TRANSACT reference architecture that are responsible for performance management. The *Operational mode manager/Coordinator components* decide to transit between operational modes, based on input obtained from the *Safety, Performance and Security Monitoring Service* components. These components monitor among other things bandwidth, availability and resource utilization and verify or predict whether application-level performance requirements (concerning e.g., response latencies) are met. When these requirements are (predicted to be) violated, the *Operational mode manager/Coordinator components* compute the optimal configuration parameters settings (i.e. optimal mode) to avoid the violation of performance requirements.

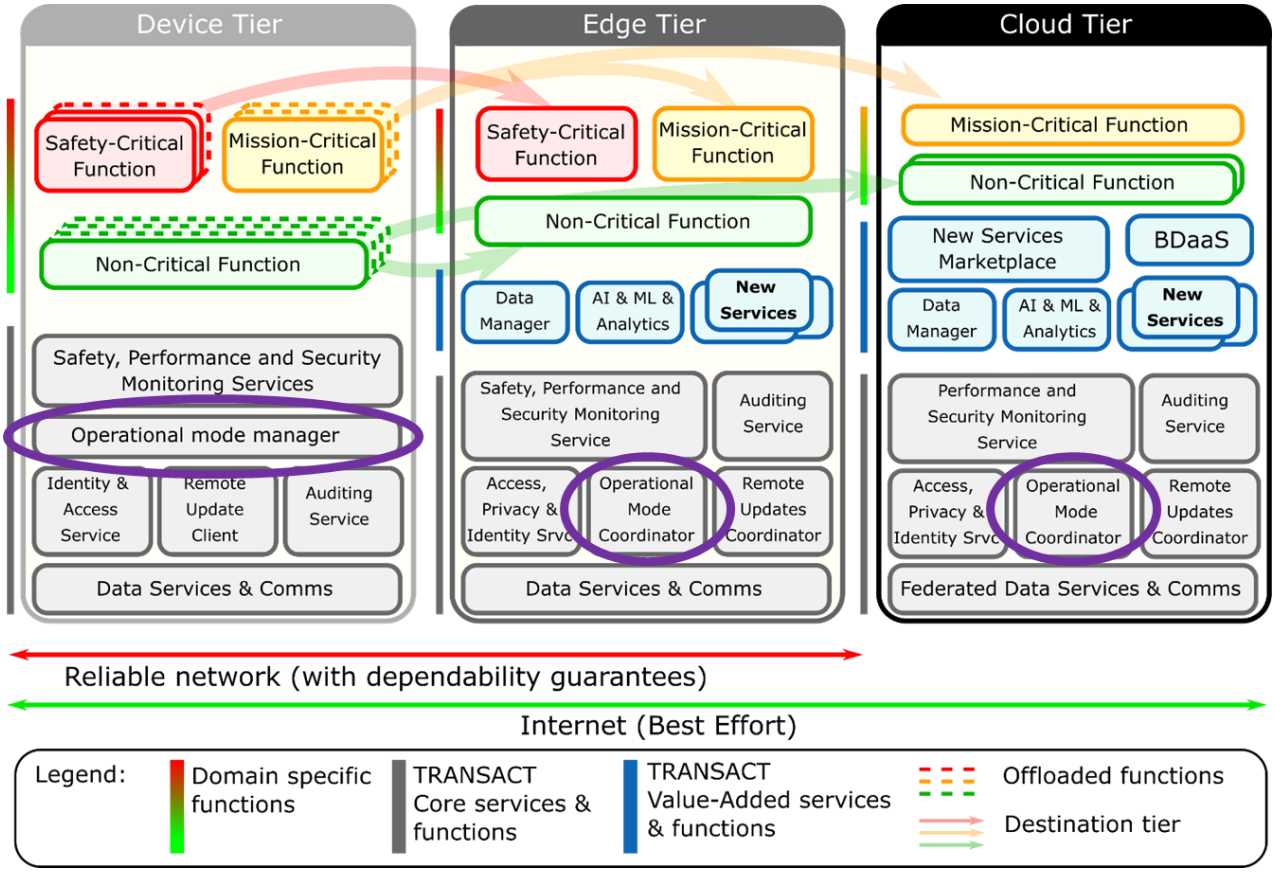

Figure 32: Performance management in TRANSACT reference architecture

Example in context of a use case

The need for performance management can be illustrated by the use case scenario "Automatic image registration" of UC4. This scenario is concerned with the ability to integrate pre-interventional 3D CT images with the live X-ray image guidance so that a cardiologist can perform the required treatment on the patient. Upon request, the system must be able to perform the image registration in the cloud, with a response time of at most 5 seconds. As a fall-back scenario of automated registration, the cardiologist should be able to perform a manual registration.

Performance management has an important role in this use case. The *Operational mode manager/Coordinator components* can adjust resource allocation or balance the workload, to prevent violation of the 5-seconds response-time requirement. In case of violation, it should transition to an operational mode of manual registration.

Challenge for application within TRANSACT context

Performance management in TRANSACT requires

- Identification of adjustable parameters (knobs) of resources and qualities.
- Identification of operational modes and (re)configuration options of the distributed platform, of the application mapping to the platform and of the application.
- Modelling: Application and platform components have configurations that influence performance in relation to other objectives, such as image quality or safety, to other costs such as cost of operation

and to the functional requirements of the application. Active management requires adequate modelling of the available configurations and corresponding relations, including the properties of the reconfiguration itself (impact on performance, quality, safety, cost, etc.).

- Optimization: Knowing the available configuration options then enables the selection of operation mode and configuration. The available options may quickly grow and the (distributed) selection of optimal mode and configuration is a challenge to be addressed.
- Reconfiguration: Strategies for actively deciding on (re)configuration, based on input acquired from monitoring.

6.5 Concepts for safety and risk analysis

This section describes selected design-time methods and concepts for analysing safety and risks to the intended system in the device-edge-cloud continuum. Safety engineering (Safety Engineering, 2022) has traditionally been concerned with assuring that engineered systems provide acceptable levels of safety. Traditional methods include failure mode and effects analysis (FMEA), fault tree analysis (FTA), and also Bow Tie analysis (Ferdous, Khan, Sadiq, Amyotte, & Veitch, 2013). These methods typically look at failure modes for each piece, part, or component of the system.

When undesired interaction between otherwise fine components, or specific scenarios, can cause unsafe situations other methods are more suited. Three novel methods and concepts are selected in this concept class which are the following:

- System Theoretic Process Analysis (STPA),
- Identification and Quantification of Hazardous Scenarios,
- The MAGERIT method for risk assessment.

The following sub sections each describe one of these methods or concepts together with their fit with the TRANSACT reference architecture, with an example in context of a use case, and the challenges still to be addressed.

6.5.1 System Theoretic Process Analysis (STPA) for safety critical Cyber-Physical Systems

Overview

STPA (Leveson, 2016) is a hazard analysis technique which not only includes component failures but also considers accidents that can be caused by unsafe interactions of system components, none of which may have failed.

In case of safety, the traditional hazard analysis is based on the decomposition of a system into separate components with the assumption that analysing each of these components separately suffices for an understanding of hazards in the overall system.

STPA is based on the idea in systems theory that, if emergent properties arise from individual component behaviour and from the interactions among components, then to control the emergent properties, such as safety, security, maintainability, and operability, requires controlling the behaviour of the individual components and the interactions among the components (Leveson, 2016).

Some of the advantages of STPA over traditional hazard/risk analysis techniques are the following:

- STPA can be started early in a concept analysis to assist in identifying safety requirements and constraints. As the design is refined and more detailed design decisions are made, the STPA analysis is also refined to help making more and more detailed design decisions.

- STPA includes software and human operators in the analysis, ensuring that the hazard analysis includes all potential causal factors in losses.
- STPA provides documentation of system functionality that is often missing or difficult to find in large, complex systems.
- STPA can be easily integrated into the systems engineering process and into model-based system engineering.

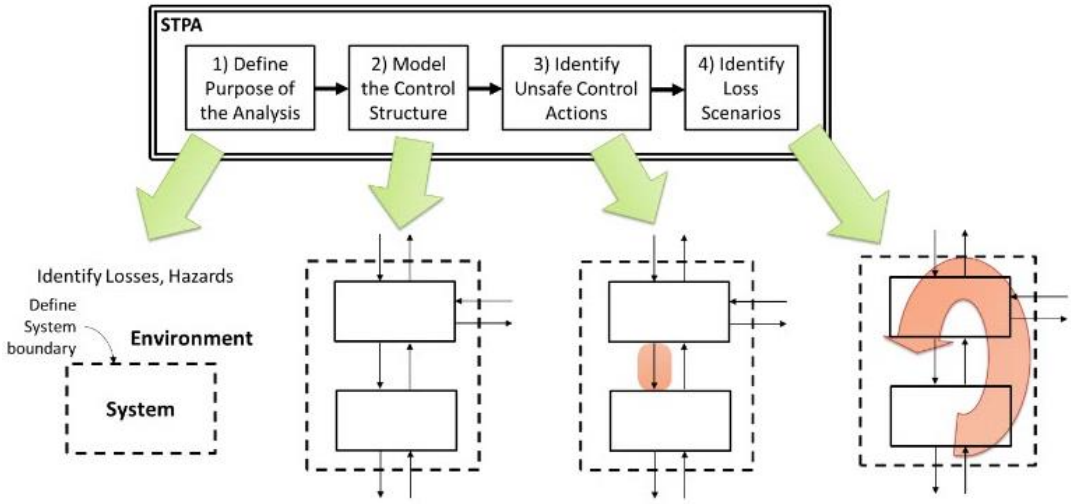

Figure 33: Overview of STPA concept (Leveson, 2016)

Description

Depicted in Figure 33 is the high-level steps in carrying out of STPA analysis. Each step constitutes of some sub-steps which provide inputs for subsequent steps, see e.g. (Leveson, 2016). The goal of the overall process is to obtain a system overview where loss scenarios are identified and can be traced back to unsafe control actions which caused them.

Fit with concept TRANSACT reference architecture/components

The end point of safety in cyber-physical systems is at the physical layer. In case of the TRANSACT project this includes the device side of the Reference architecture. However, as we outlined in previous sections, safety is an emergent behaviour of a system. This implies to ensure safety all the elements that are included in the solution should be considered and reasoned about. Having this in mind, means that safety analysis is a cross-cutting concept. In Figure 34, the blue arrows highlight that the line of reasoning about safety affects all tiers involved in the provided solution.

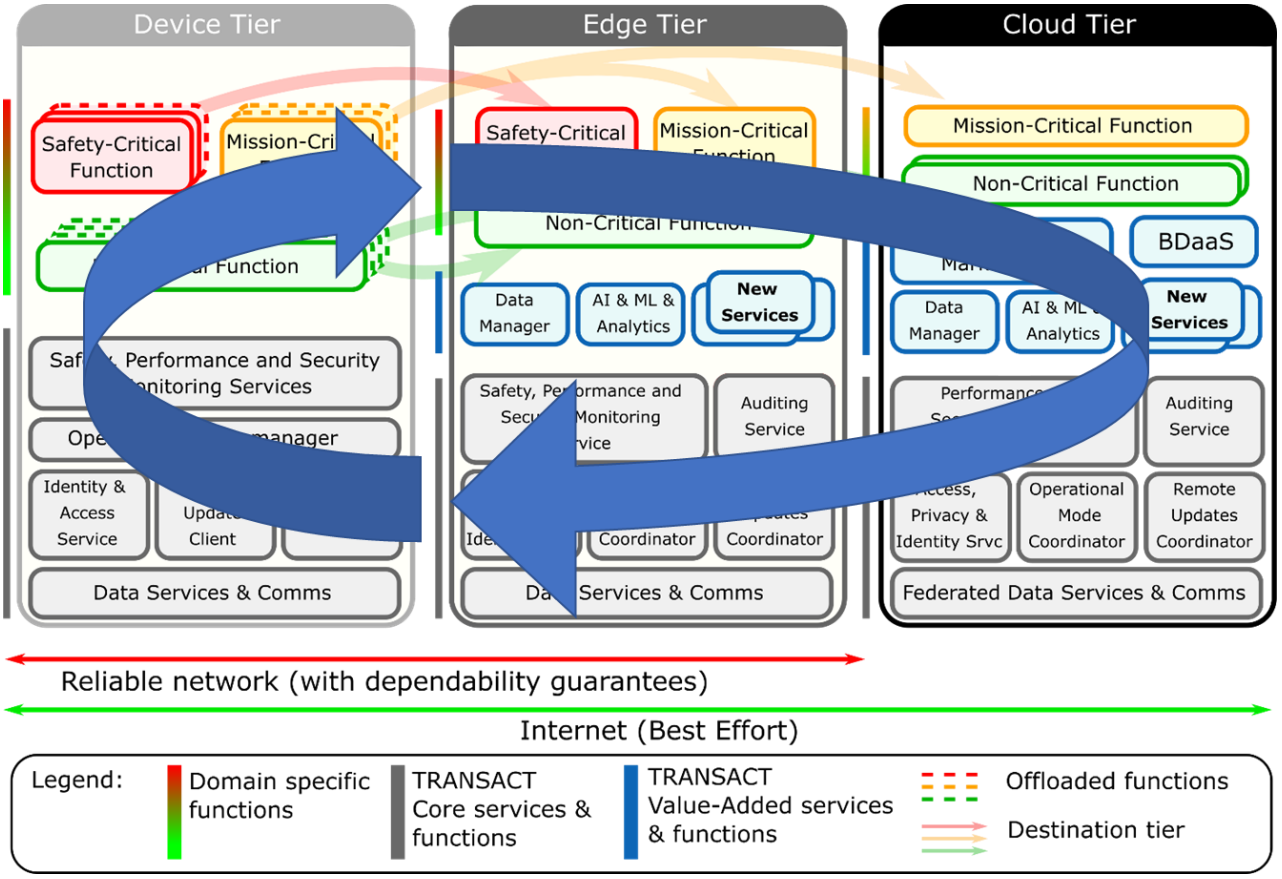

Figure 34: TRANSACT reference architecture

Example in context of Use case 4 (model registration service)

To illustrate how STPA can be applied in context of TRANSACT project, use-case 4, Image guided therapy and diagnostic imaging systems, is chosen (see Section 3.1.4). More specifically, the cloud-based image-registration scenario is considered, in which calibration of the overlay of the live image on top of a pre-computed 3D model is calculated in the cloud.

1. **Define Purpose of the Analysis**

To define the purpose of the analysis we need to define the system level losses by focusing on what safety or mission critical situation can arise within the context of the system. Following is the example of Loss in the context of cloud-based image-registration of use case 4.

L-1. Loss of life or serious injury to people
L-2. Loss of mission (success of operation/treatment)
L-3. Loss of performance (slow/hick-ups/…)
L-4. Loss of customer satisfaction

Apart from Losses the Hazards in the context of system operation are identified, e.g.

H-1. System Registers wrong patient [L-1, L-2, L-4]
H-2. System is unable to connect to cloud [L-2, L-3, L-4]
H-3. System streams live images on dedicated monitor with a lag [L-1, L-2, L-3, L-4]

2. Model the Control Structure

In this step a hierarchical control structure is modelled that consists of feedback loops. Figure 35 shows an example of such a control loop. In this figure a cloud-connected hospital operating room is depicted in which a surgeon and team operate on a patient with an Image-guided therapy system (controller and controlled system) which requests a image registration service from the cloud. At both hospital and cloud side operational mode manager and coordinator are in place sync the procedure.

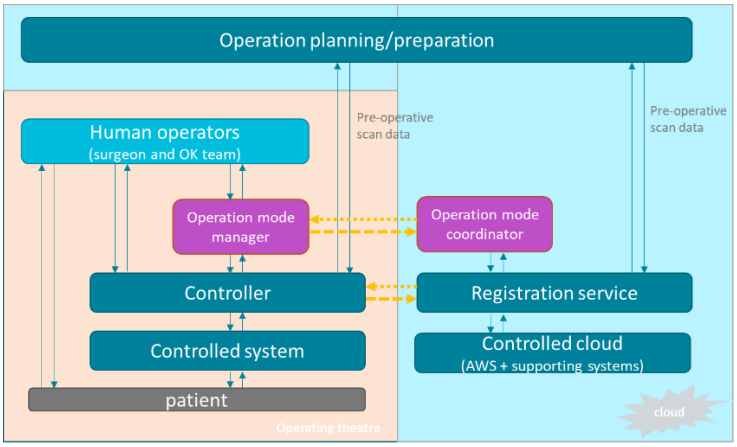

Figure 35: Example model of control structure for the UC 4 cloud-based image-registration scenario.

3. Identify Unsafe Control Actions

After modelling the control structure, the Unsafe Control Actions (UCA) are identified. An unsafe control action may lead to a hazard scenario in a worst-case scenario,. Four principle ways are distinguished in causing an control action to be not safe (Leveson, 2016)

1. Not providing the control action leads to a hazard.
2. Providing the control action leads to a hazard.
3. Providing a potentially safe control action but too early, too late, or in the wrong order
4. The control action lasts too long or is stopped too soon (for continuous control actions, not discrete ones).

For example, in the use case 4 cloud-based image-registration scenario, the following unsafe actions could potentially occur during operation, before safeguards would be in place (see Table 3):

Table 3: Example analysis of potentially unsafe actions

ACTION	NOT PROVIDING CAUSES HAZARD	PROVIDING CAUSES HAZARD	TOO EARLY, TOO LATE, OUT OF ORDER	STOPPED TOO SOON, APPLIED TOO LONG
Operation	….	Human operator operates based on another patient's data [H-1]	Human operator continues to engage in operation while the live image suffers from lag in the overlay. [H-3]	….

Figure 36 shows the analysis control flow to derive scenarios could cause an human operator to potentially perform an unsafe action (e.g. the example unsafe actions as indicated in Table 3).

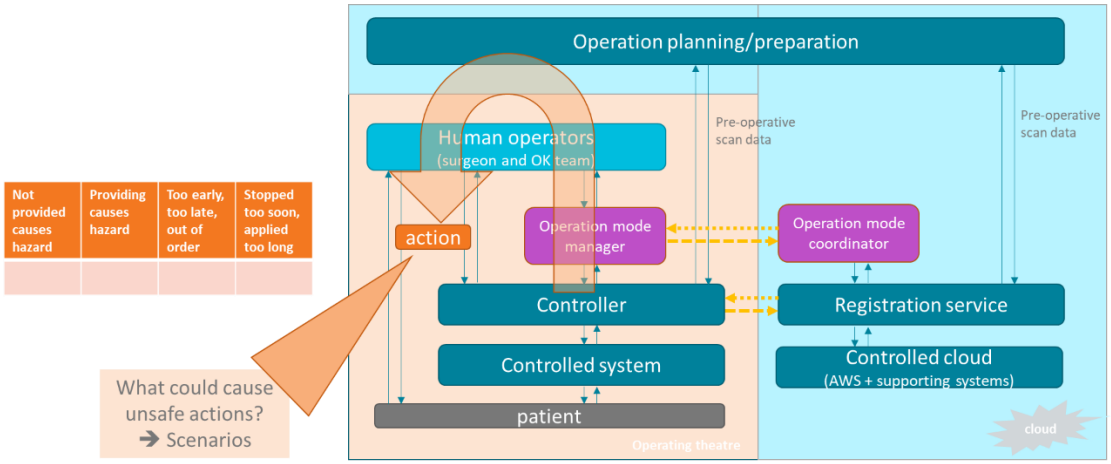

Figure 36: Analysis of potentially unsafe actions yield loss scenarios

4. Identify Loss Scenarios

A loss scenario describes the causal factors that can lead to the unsafe control actions and to hazards. To identify the cause two questions can be asked (Leveson, 2016):

a) Why would Unsafe Control Actions occur?

b) Why would control actions be improperly executed or not executed, leading to hazards?

In the use case 4 cloud-based image-registration scenario, the following scenarios are identified:

Scenario-1. For UCA-1: Wrong registration of patient data results in incorrect imagery without indicating to the human operator who carries on the operation [UCA-1] based on wrong data. [H-1]

Scenario-2. For UCA-1: Due to lag in live stream overlay of the operation without alarming the human operator, the operator continues the operation [UCA-2].

Challenge for application within TRANSACT context

Although STPA is a general concept, the specificity of TRANSACT project requires further elaboration to make it fit for the distributed Cyber-Physical Systems. Additionally, it needs to be assessed how the 3-Tier architecture would affect the 4-step process in STPA. Specific points of attention are the following:

- Scalability of application and platform affects the system/component decomposition. The open question is how this would affect the STPA method.
- The serverless architecture in some of the applications remove the need to focus on the hardware. Nonetheless this can have tremendous impact on safety related issues.

6.5.2 Identification and quantification of hazardous scenarios

Overview

The Automation Risks Method is an integrated method for safety assessment of automated driving functions which covers the aspects of functional safety and safety of the intended functionality (SOTIF), including identification and quantification of hazardous scenarios. The proposed method uses and combines established exploration and analytical tools for hazard analysis and risk assessment in the automotive domain, while adding important enhancements to enable their applicability to the uncharted territory of safety analyses for automated driving.

The method is tailored to support existing safety processes mandated by the standards ISO 26262[(ISO, 2011)] and ISO/DIS 21448 [(ISO, Under development)] and complements them where necessary. It has been developed in close cooperation with major German automotive manufacturers and suppliers within the PEGASUS project (PEGASUS Project, 2022).

In the following we will briefly describe the steps involved in the suggested method which is depicted in Figure 13. The method is split into the Identification and Quantification of Hazardous Scenarios. Detailed information about the method and the background can be found in [(Kramer, et al., 2020)].

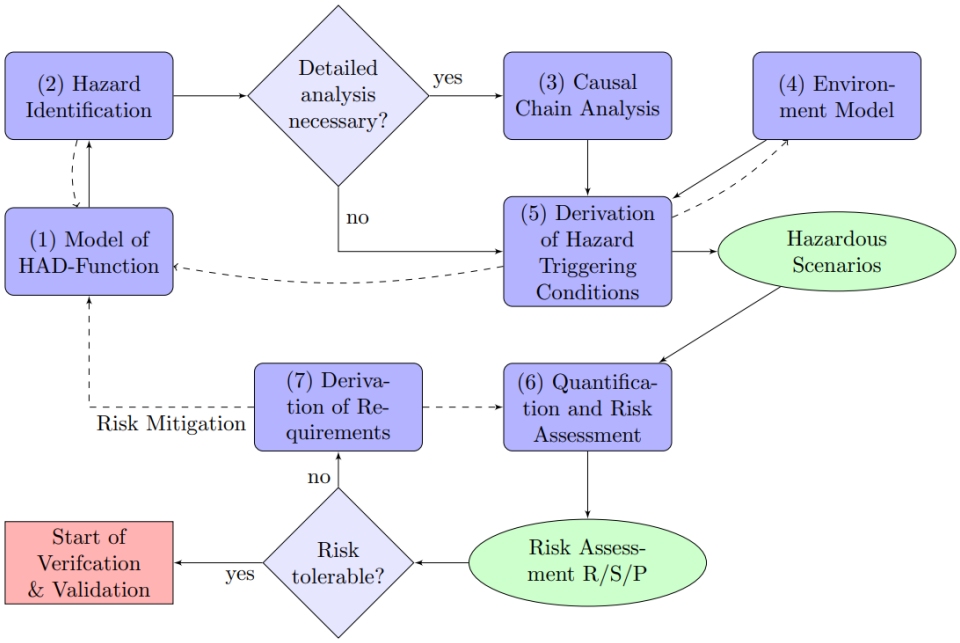

Figure 37: Identification of critical scenario properties

Method for Identification of Hazardous Scenarios

Hazard Identification

Input:

- Model of the Automated Driving System (ADS) which specifies the functional architecture and the intended functionality (Item definition)
- Model of the Operational Design Domain (ODD)
- adjusted list/catalogue with key words
- (optionally) catalogue/ data base with criticality phenomena (structured knowledge of hazards in the ODD)
- relevant hazards for similar systems or system classes (e.g. accident reports)

This step aims at identifying hazards caused by performance limitations or functional insufficiencies and their possible triggers in the environment via an adapted Hazard and Operability Studies (HAZOP) ((IEC), 2002) approach.

Output:

- List/ Catalogue with identified hazards and functional insufficiencies for the investigated system

Causal Chain Analysis

Input:

- List/ Catalogue with identified hazards and functional insufficiencies
- concreted item definition
- optionally real-world data

This step aims at investigating causes rooted in the system and triggering environmental conditions of all identified hazards for all basic scenarios via extended fault tree analysis. The goal is to identify all combinations of triggering environmental conditions.

Output:

- conditional qualitative fault trees for all identified hazards and basic scenarios

Derivation of hazard triggering scenario properties

Input:

- qualitative fault trees
- Model of the environment (with regard to the ODD) in a scenario specification language

This step aims at transferring the conditional fault trees into associated (abstract) scenarios by reducing the fault trees to their environmental conditions, translating these conditions in an appropriate scenario language and classifying them temporally.

Output:

- reduced qualitative fault trees with formalized environmental conditions
- catalogue of hazards and for each hazard a set of scenarios, consisting of logically connected and temporally classified environmental conditions

Method for Quantification of Hazardous Scenarios

Quantification & Risk assessment

Input:

- reduced qualitative fault trees with formalized environmental conditions
- probabilities of occurrence and error rates of relevant environmental conditions

This step aims at the quantitative assessment of the derived scenarios e.g., by determining the probabilities of occurrence, controllability, potential severity, prioritizing the scenarios due to their relevance and deriving a risk assessment.

Output:

- quantitative fault tree
- risk assessment for identified hazards
- characteristic scenarios describing the emergence of hazards

Derivation of requirements

Input:

- risk assessment for identified hazards
- characteristic scenarios describing the emergence of hazards
- regulatory guidelines and requirements

This step aims at checking if regulatory guidelines and requirements are complied with and in case of doubt how to reduce the calculated risk.

Output:

- requirements for error rates or exposures
- Suggestions of risk mitigating measures

Fit with concept TRANSACT reference architecture/components

The end point of safety in the cyber-physical Systems manifests at the physical layer. In terms of the TRANSACT project the safety critical functionality is deployed to the device tier. For reasoning about the overall safety, the whole system with all involved components has to be considered. To ensure safety, all elements involved have to be considered during the safety analysis. This makes safety analysis a cross-cutting concept. In Figure 38 we highlight that the process of safety analysis affects all tiers in the provided solution.

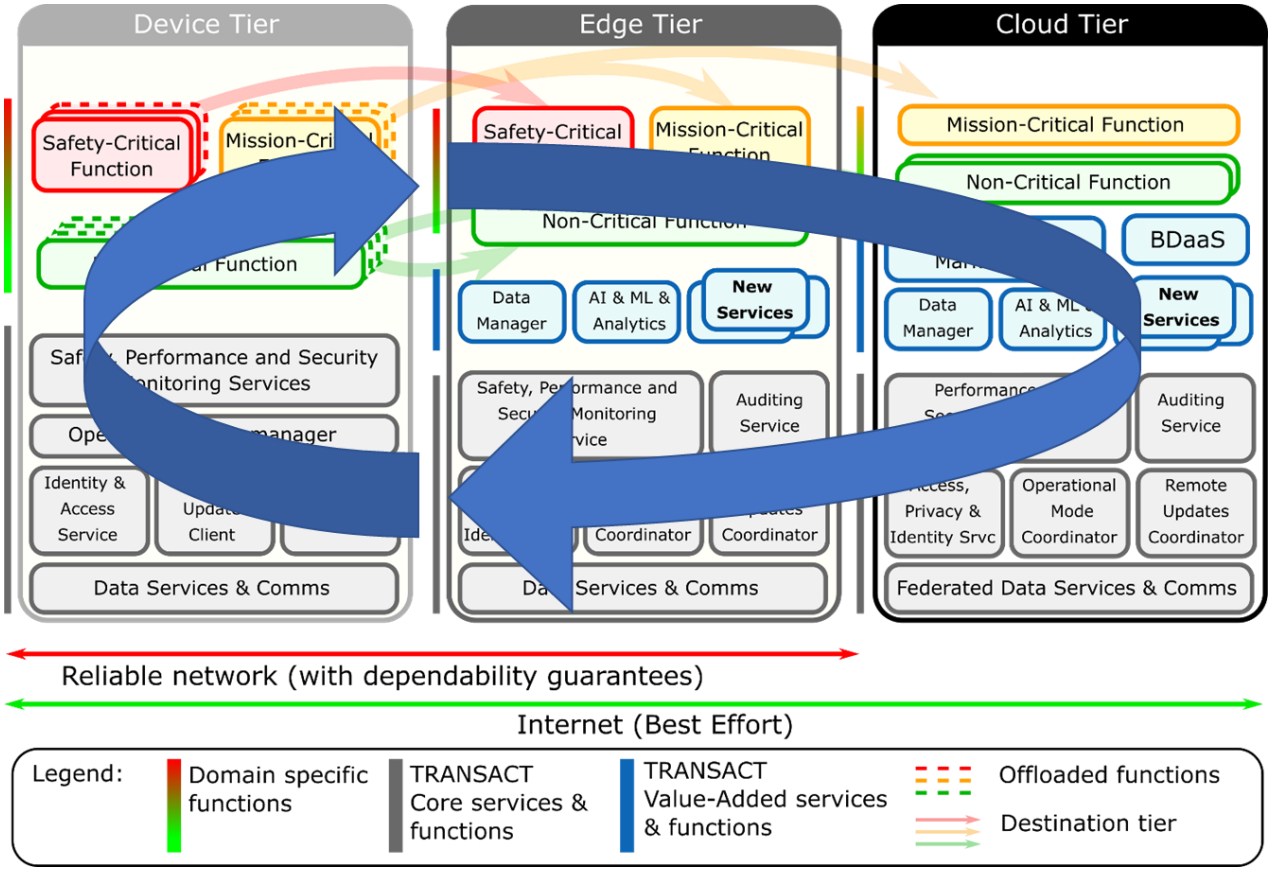

Figure 38: Identification of Hazardous Scenarios in the TRANSACT reference architecture

Example in context of a use case

The identification of critical scenario properties is of especially high importance in Use Case (UC) 1 and UC 2. These use cases are operating in a context were a scenario-based verification approach is necessary. In the context of a scenario-based Verification & Validation (V&V) process it is infeasible to simple simulate every

possible scenario of these systems. It is well known that in the context of Use Case 1 (automotive systems) a mileage-based approach is infeasible [(Kalra, 2016)]. Therefore, it is a central challenge to systematically identify and quantify scenarios that are likely to exhibit hazardous behaviour of the ADS. Identifying hazard-triggering environmental conditions during the safety analysis helps to reduce the set of possible scenarios to a manageable set of relevant scenarios.

Challenge for application within TRANSACT context

Although the proposed method is designed in a domain agnostic way as a general concept, the specificity of the TRANSACT project requires further elaboration to make it fit for distributed Cyber-Physical Systems. We need to further investigate how the effect of distributing functionality of a system to the cloud is reflected by the method.

Additionally, some tailoring is needed to make it fit for the maritime domain of Use Case 2. Especially in regard of the used model of the operational domain and the language for describing the scenarios.

6.5.3 The MAGERIT methodology for risk assessment

Overview

MAGERIT methodology is a standard that establishes principles for the effective, efficient and acceptable use of IT. It has been prepared by the CSAE (Spanish Higher Council of E-Government) and published by the Ministry of Finance and Public Administrations (HIGHER COUNCIL FOR ELECTRONIC GOVERNMENT). This methodology is well known and recommended in Europe by ENISA (the European Union Agency for Cybersecurity) (MAGERIT, 2005).

The MAGERIT methodology (Methodology of Analysis and Management of Risks of Information Systems) has as objective to help organisations balance risks and encouraging opportunities arising from the use of IT. Risk analysis allows to know the information systems: their assets, their value, and the threats to which they are exposed, moreover, it provides a balanced framework for Governance, Risk Management and Compliance, in order to prevent conflicts and threats.

Figure 39 shows the steps in the MAGERIT methodology. The figure shows how, first, it is needed to identify the assets that form part of the system. Assets are exposed to several threats and vulnerabilities which occur with higher or lower frequency. When an incident occurs, it may impact and potentially depreciate the asset(s) affected by it, causing a certain impact. Despite the unfortunate event, if we figure out the probability of materialising the threat, we could conclude the risk in the system or the loss to which it is exposed. On the other hand, depreciation and probability qualify the vulnerability of the system to a threat. Finally, with this assessment it is possible to set safeguards and limit the already identified risk up to a residual or acceptable risk value.

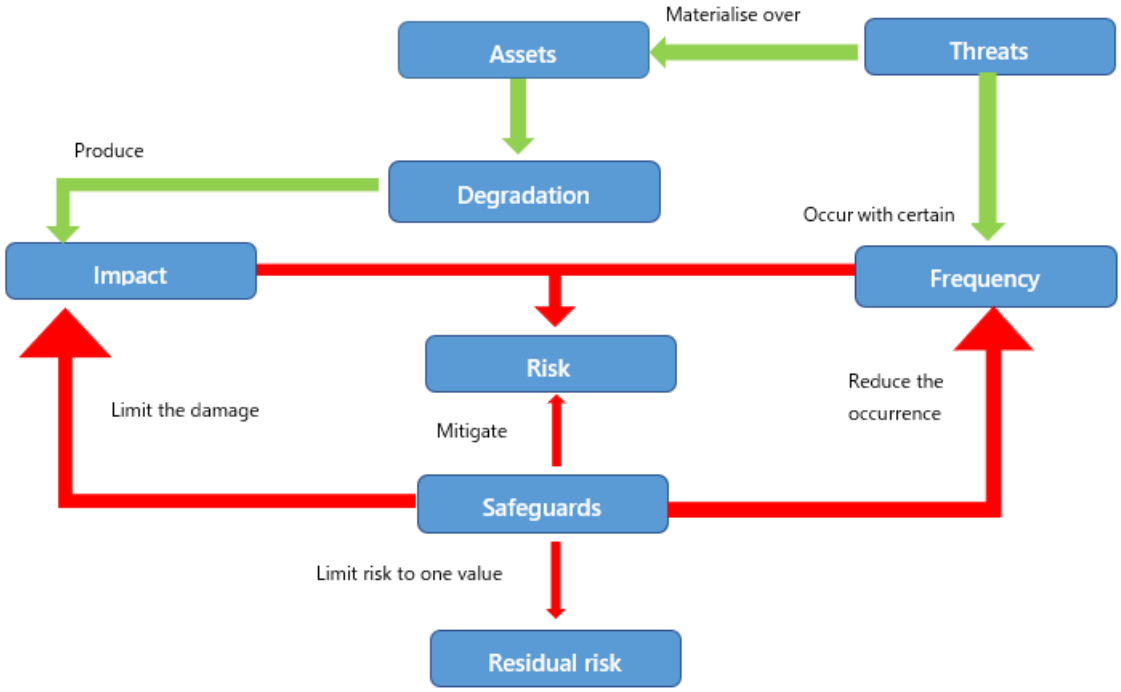

Figure 39. The MAGERIT methodology

Once the most critical risks are known, the entity can deploy safeguards or controls to deal with threats. Safeguards mitigate the impact and risk values to residual values. Residual values mean the risk and impact that remains after putting effort to identify and eliminate or mitigate some or all types of risks previously identified. Figure 40 summarises the process of risk analysis following the MAGERIT methodology.

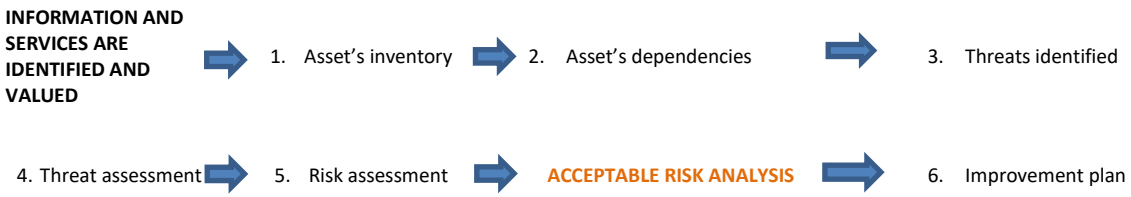

Figure 40. Process of risk analysis.

Fit with concept TRANSACT reference architecture/components

In Figure 41, the purple ellipses highlight the line of reasoning about safety is recursive across all the tiers involved in the provided solution. So, in here it is shown how *Safety, Performance, and Security Monitoring Services* is a core service in Transact architecture. The purple ellipses show the components on three tiers of the Transact reference architecture that are responsible for performance management.

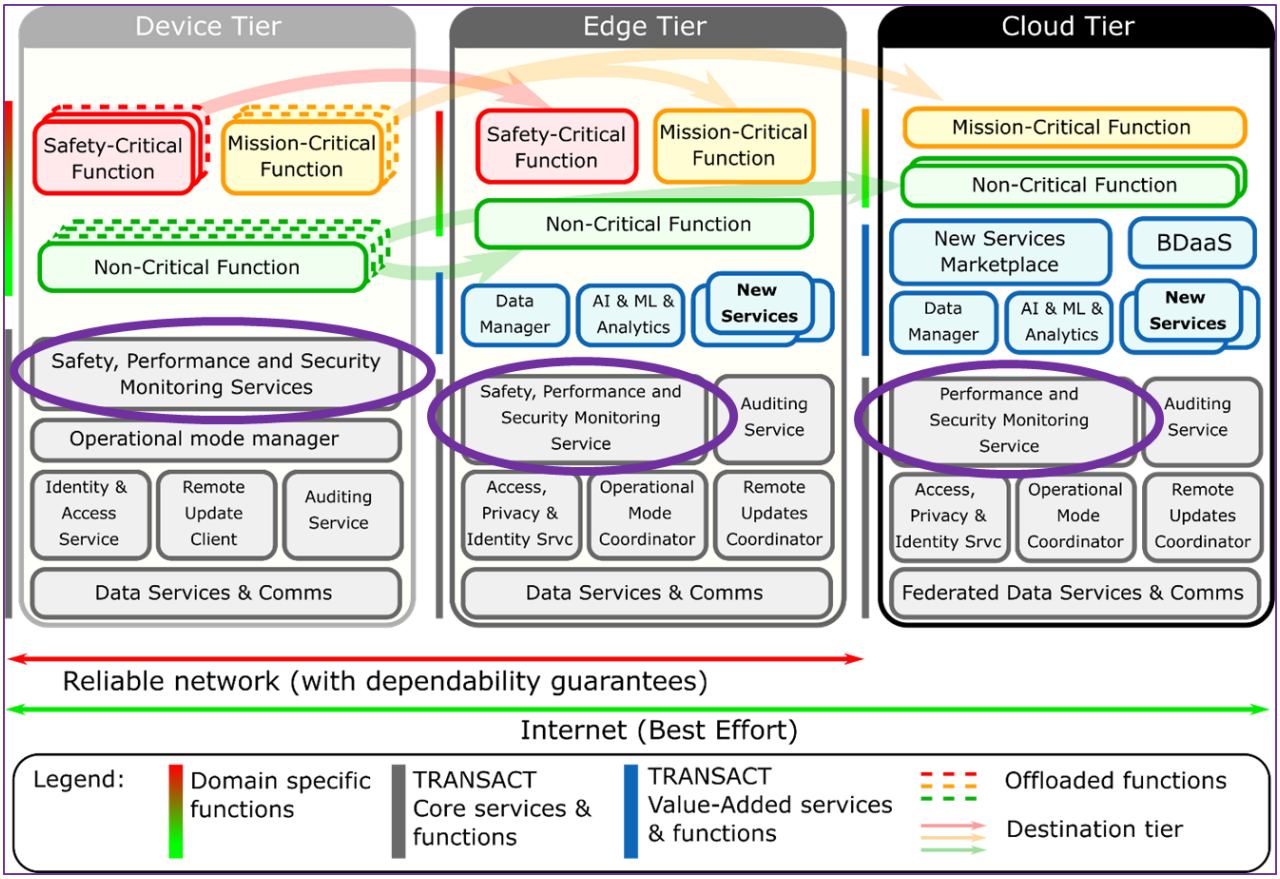

Figure 41. Scope of the MAGERIT methodology in the TRANSACT reference architecture

In order to achieve the goal of performance management, it is necessary to set monitoring, optimisation, and protection of the industrial processes applying Big Data and Artificial Intelligence.

For carrying out the risk analysis itself, the platform GConsulting can be used. This tool is based on the international standard on how to manage information security ISO/IEC 27001. In addition, in order to a wider perspective, this is not the only reference that will be use. Instead, to succeed in a complete risk analysis, both, MAGERIT and GConsulting integrate the Spanish *Royal Decree 3/2010, 8th January, of regulation of the National Security Framework*, which performs on establishing the policy of security in the use of electronic means, setting the principles and requirements that adequately ensured information security treatment.

Although MAGERIT is specialised in Information and Communication Technologies, it covers also Safety, Performance, Security and Privacy.

Example in context of a use case

The need for implementing a risk analysis can be illustrated by the use case scenario *"Transformation of the monolithic critical system in wastewater treatment plants to the distributed system supported in the cloud: management of the biological reactor"* of UC5. This scenario is concerned with controlling the process and improving water quality. As risk analysis allows to know which work elements are subject, a wastewater treatment plants analysis will enhance the analysis and obtention of insights by the operator and lead to newer and more advanced applications (predictive maintenance) that will result in a reduction of downtime, costs, and better service. Setting a risk management procedure may facilitate governing bodies to take decisions with explicit consideration of the risks derived from the use of IT and AI.

Risk analysis has an important role in this use case. The *Safety, Performance, and Security Monitoring Services* can identify the potential threats involved (*e.g.,* a spill or a natural disaster) so an unfortunate event can be avoided, or, at least, the potential damages may be diminished.

Challenge for application within TRANSACT context

Risk management in TRANSACT requires potential extensions or adaptations for five completely different use cases of device-edge-cloud continuum systems. Hence, a thoroughly research should take place, in collaboration with the partners. Finally, a common challenge when managing risks, is also to raise awareness among the responsible persons of the importance of taking risks into account.

6.6 Concepts for health and safety monitoring

This section describes selected run-time methods and concepts monitoring the safe operation of the device-edge-cloud continuum. When safety-critical or mission-critical functions are offloaded to the edge or cloud, then monitoring at run-time is necessary to ensure that the system is still operating within specifications (and is safe to use). Health/safety monitoring can encompass the monitoring of availability of the device, edge, cloud, liveness, responsiveness and integrity of applications.

In this section two monitoring concepts are described:

- SIRENA concept and platform for monitoring operation and network behaviour
- Health dependency graph concept for monitoring application logic and sensors/signals

The following subsections each describe one of these concepts together with their fit with the TRANSACT reference architecture, with an example in context of a use case, and the challenges still to be addressed.

6.6.1 Concept and SIRENA platform for monitoring operation and network behaviour

Overview

SIRENA is an industrial software solution for network management that implements a concept for monitoring operation and network behaviour and provides support and solution for distributed environments. In a preliminary phase, it assesses the architecture, edge and cloud services, and how to leverage them.

Installing a low-level probe, SIRENA sniffs the traffic passively (without altering the network) on a real-time network. Firstly, NOZOMI generates a topology and identifies each type of device. Then, in the real time network the operator can select a specific device or a group of devices and make a snapshot that shows the specific traffic, in order to do a machine learning study of the behaviour and do a future monitoring and detection of unwanted behaviour.

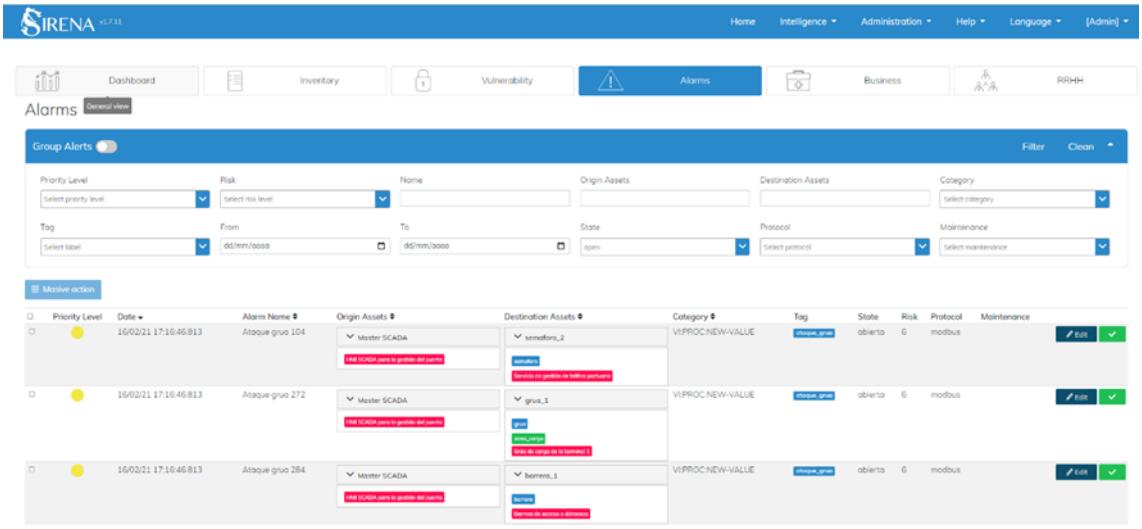

Figure 42: Alarms in SIRENA

In order to measure the severity of the potential threats detected, a Risk Analysis should be carried out of the complete system which it will be correlated with the Ip´s / Mac number of the devices (IT and OT).

This methodology is used inside Critical Infrastructures (such as Hospitals, harbour environments and Electrical providers) and also among industrials networks.

In the industrial control systems (ICS) it is necessary to know the operation of industrial processes and their particularities in terms of protocols (Modbus, Profibus, IEC 104, KNX, Ethercat, etc.) and industrial devices such as RTUs, IEDs or PLCs.

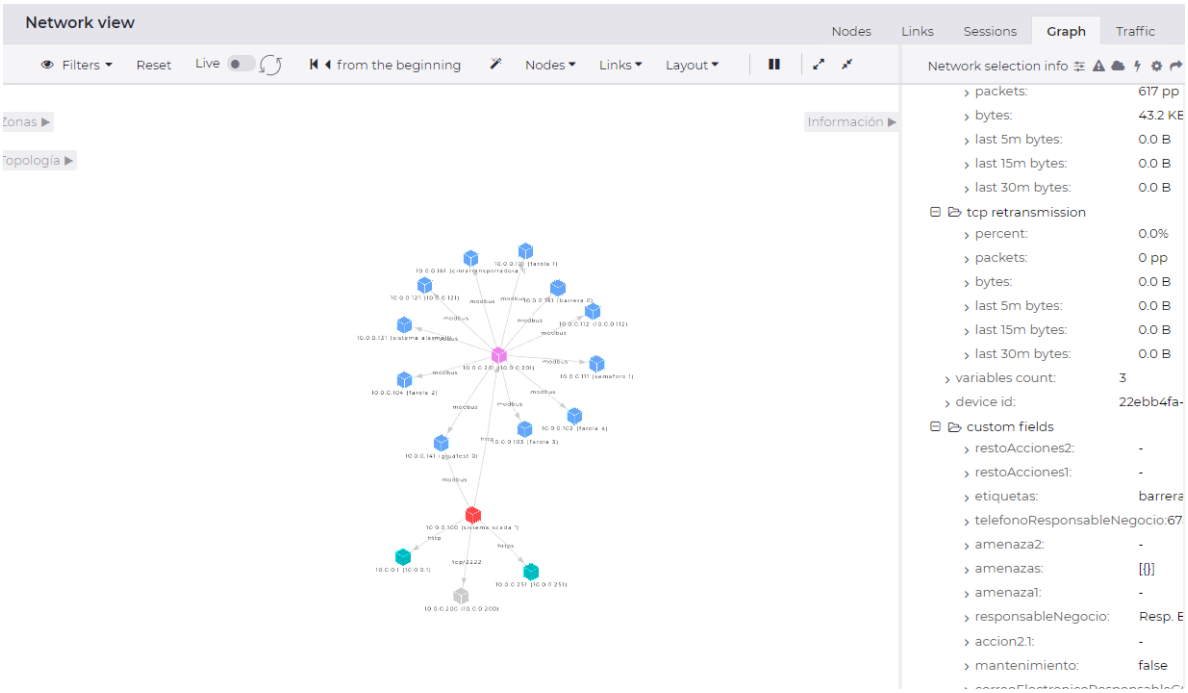

Figure 43: Network map of assets and interconnections in NOZOMI

Use of the SIRENA platform guarantees the availability and continuity of business, industry and critical infrastructures. SIRENA works with NOZOMI (a low-level probe), and focuses on detecting cyberthreats and vulnerabilities by studying the behaviour of the network. SIRENA permits a real-time monitoring and identification of potential abnormal behaviours. The result of such an analysis allows a faster response to attacks, and improves resilience of the system.

After the machine learning phase, the system has a baseline of the desired or normal behaviour and enters into a detection mode, during which alerts will be triggered based on different patterns:

1. Detection of known attack patterns or undesired behaviour of each technology.
2. Detection of anomalous behaviour outside the baseline: Variables out of range, cycles different from normal, change in the orders sent by SCADA, etc.
3. Manually programmed alerts for detection of patterns or sensitive values.

Nozomi's SCADA Guardian supports the management of security events through a correlation engine and a detailed collection of tools, which serve to have an overview of all security-related aspects of your infrastructure.

Furthermore, this system is a plug-and-play installation without blocking any of the industrial installations: it is a device (physical or virtual) that connects to the industrial network in a non-intrusive and completely passive way.

Additionally, this system provides a real-time view without affecting any network: it studies to all the traffic on the control and field networks, analysing it at all levels of the OSI stack (from L1 to L7).

Finally, there is an automatic profiling of devices, functions and their level of risk: it uses machine learning techniques to create a detailed behavioural profile for each device according to the state of the process, detecting critical states.

SIRENA concept and platform technology key points:

Security:

- Enables compliance with regulations such as PIC Law (Critical Infrastructure Protection), ENS (National Security Scheme), etc.
- Detects faults in the industrial network (commands, variables, poorly implemented protocols), in order to monitor the correct implementation of integrators.
- Detection of security threats, since SCADAGuardian analyses traffic and detects patterns that reveal attacks (it can also be fed with SNORT rules from other systems).
- Detection of anomalous behaviours, since SCADAGuardian establishes a baseline of good performance, and detects anomalies that may reveal failures, intrusions, unauthorized changes.

Safety:

- Discovery of industrial networks and their functioning. Self-inventory in project and pilot.
- Common language and understanding between operations and IT staff.
- Visibility to IT staff to IT operations (in a non-intrusive way).

Fit with concept TRANSACT reference architecture/components

In Figure 44, the purple ellipses indicate the components responsible for the Safety, Performance and Security Monitoring Services. SIRENA can be positioned as part of all these monitoring services in all tiers.

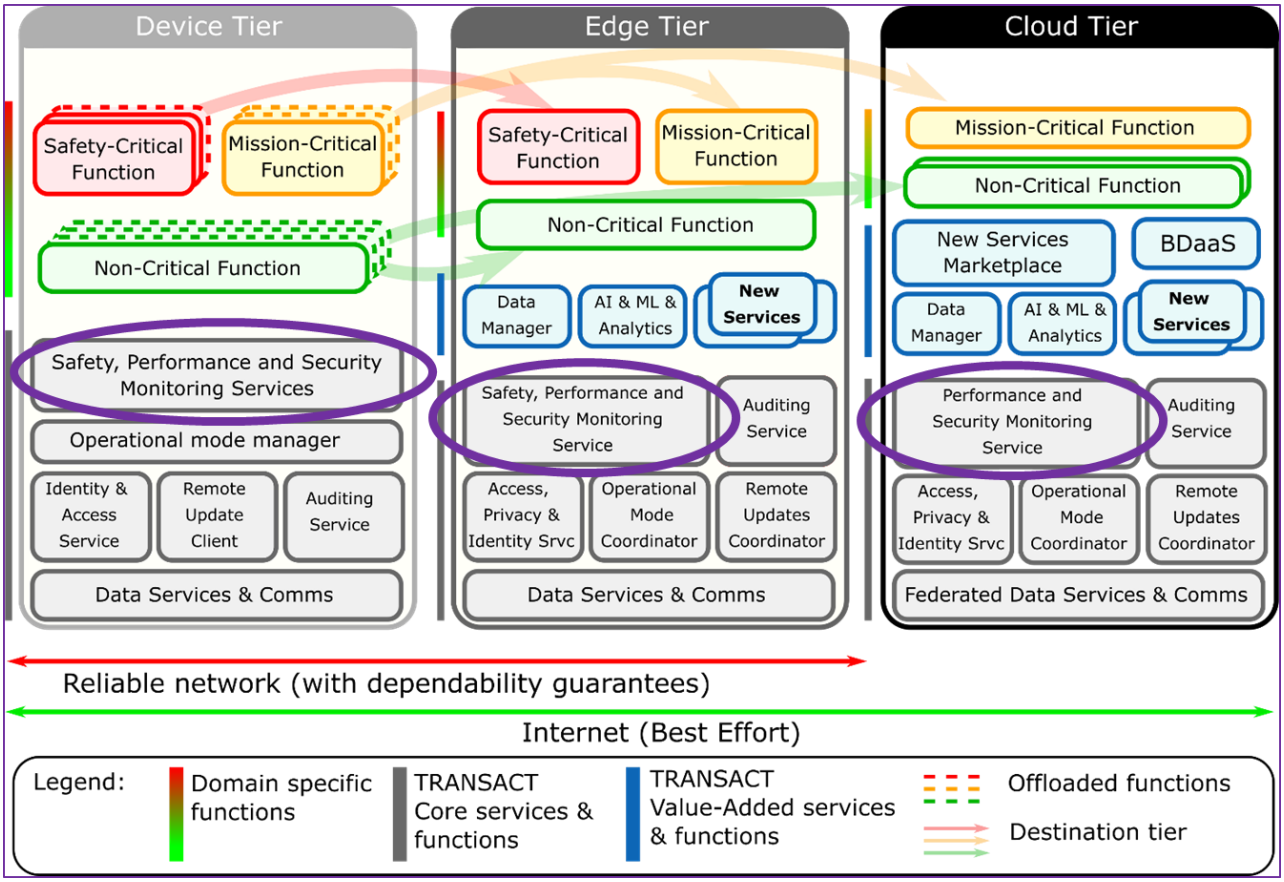

Figure 44: SIRENA positioned in the TRANSACT reference architecture

Example in context of a use case

RENFE, the Spanish railway company is implementing the SIRENA tool in several stations in Madrid.

Also, the Valencian harbour is developing the rules of correlation to identify and label all the assets under Port 4.0 project. Another reference in the medical sector is the 4 Hospitals in GVA (Generalitat Valenciana) where we have already tested SIRENA.

The need for monitoring operation and network behaviour can be illustrated by the use case scenario *"Transformation of the monolithic critical system in wastewater treatment plants to the distributed system supported in the cloud: management of the biological reactor"* of UC5. This scenario is concerned with controlling the process and improving water quality. Furthermore, a wastewater treatment plants analysis will enhance the analysis and obtention of insights by the operator and lead to newer more advanced applications (predictive maintenance) that will result in a reduction of downtime, costs, and better service.

Through the monitoring system any abnormal behaviour, such as a leak, or low water pressure will be detected by SIRENA and alerted to the staff.

Challenge for application within TRANSACT context

From the point of view of privacy and security in CPS distributed solutions, the innovation from TRANSACT will materialise on the "Risk Analysis" tool aligning both critical processes and the infrastructure deployment in this scenario needed to minimise the risks identified in a design stage, and design strategies and actions to avoid these risks. Applying SIRENA to the TRANSACT use cases requires:

- Developing a tool capable to manage failures in IT with OT networks.
- Monitoring the abovementioned failures and have a faster response.
- Translating the detected anomalies into human understandable terms and language.
- Adapting the system for each asset and infrastructure.
- Applying machine learning techniques to create a profile which detects critical states.
- Introduction of the concept of "Semantic Security", which means the security of an encryption scheme in CPS distributed solutions.
- Simulation of possible threats to stress and challenge the system, so SIRENA can learn from them.

6.6.2 Health dependency graph concept for monitoring application logic and sensors/signals

Overview

Components (or services) of a Cyber-Physical System need to be monitored in order to ensure that the system is functioning safely. Such components can be, for example, different sensors, edge devices or cloud services. Their correct functioning may depend on other components, i.e. a failure in one component might affect the correct operation of another component. Understanding and monitoring these dependencies hence is vital to ensure safety in a safety-critical CPS.

For monitoring purposes, the health of a single component can be described with the following three-step scale:

- *Green*: the component is functioning as it is supposed to be (according to spec).
- *Yellow*: the component is still functional, albeit with a limited capacity.
- *Red*: the component has failed, and mitigation actions are needed.

For each component, a health dependency graph describes the relationships to other components – capturing how the health of connected components affects the health of the current component and vice versa.

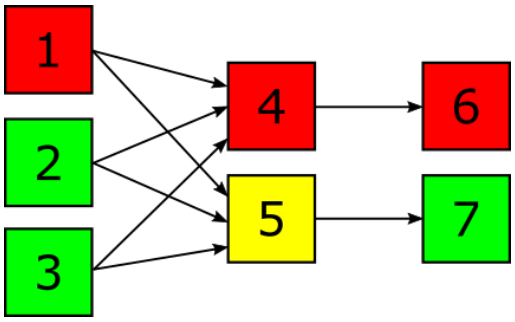

Figure 45: Example health dependency graph

Figure 45 visualises an example health dependency graph between various system components. In this figure, component 1 has failed. Component 4 operates in a configuration, where it fails if any of its dependencies (1, 2 or 3) fails. Component 5 fails only partially if at least one of its dependencies (1, 2 and 3) is still healthy. Component 7 now depends on 5, but can tolerate the component 5 being only partially

healthy. Finally, component 6 fails given its dependency on the non-healthy component 4. Using this kind of graph, we can propagate failures from sensor to application user level and between different services.

One important use case for monitoring is to detect a need for human intervention. Ideally, the system operates on its own; minor non-critical failures should be tolerated or corrected automatically by the system. However, some failures are of such nature that the system cannot resolve them own its own. In that case, human intervention is needed.

When that situation arises, a system safety monitor can be used to show that one of the components is not behaving correctly, and is requiring human attention. The operator can then try to resolve the problem (restoring the system to a healthy state), or shut down the system safely before a safety-critical hazard could arise.

Fit with concept TRANSACT reference architecture/components

The health dependency graph concept is a cross architectural concept, and can be deployed as a base concept inside all functions, inside all tiers, and between different tiers (see the green boxes in Figure 46).

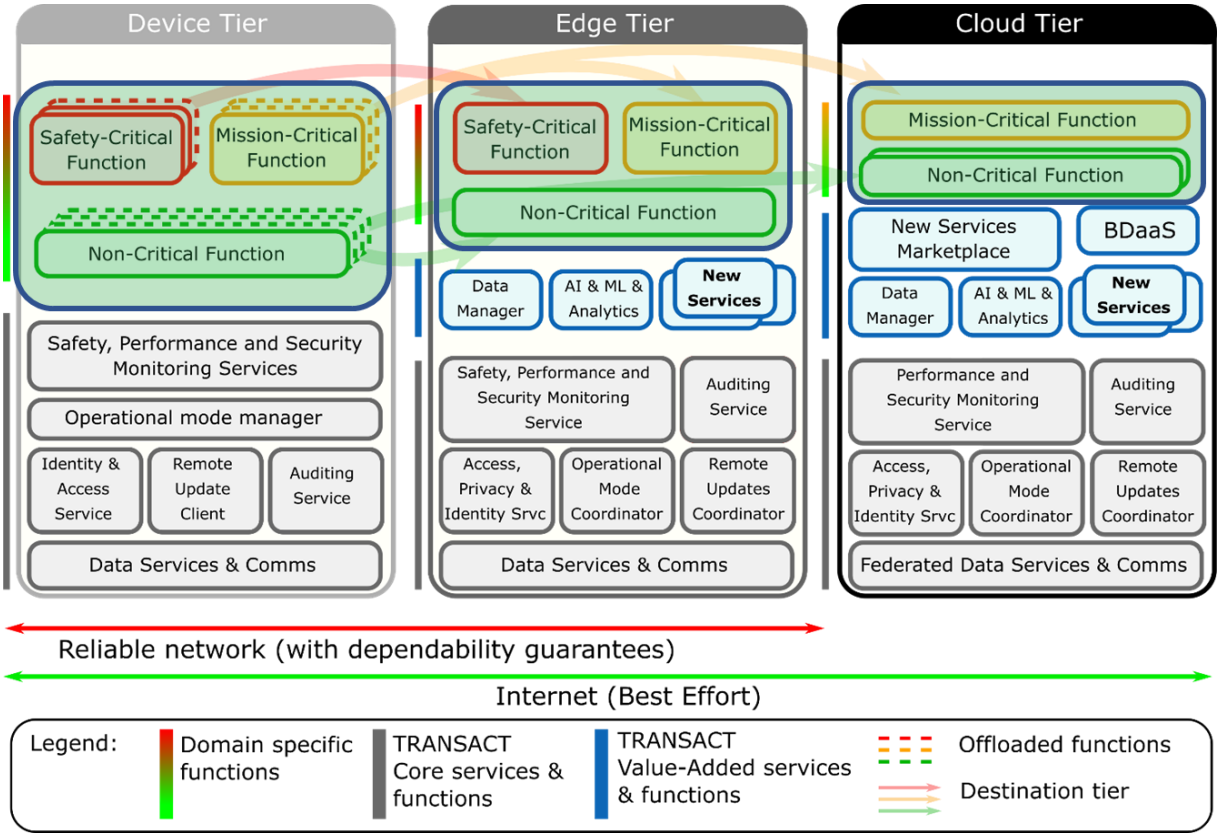

Figure 46: The health dependency graph is inside all functions of the TRANSACT reference architecture

Example in context of a use case

In UC1, autonomous vehicles could potential interpret the state of traffic lights using a camera as primary means of detection. However, looking at a traffic light is the human way to observe whether one is allowed

to drive through an intersection. Especially in a city, using cameras for direct interpretation of traffic light states is questionable: there are a lot of distractions in the urban environment that could cause a camera to misinterpret the situation. Alternatively, sending traffic light imagery to a cloud tier for interpretation, and then back to a vehicle or edge tier increases the risk for long latency. Therefore, Nodeon has chosen to gather traffic light data electronically from the traffic light itself, and convey that to the autonomous vehicle.

Nodeon uses a TLEX I2V (Traffic Live Exchange Platform) Node to gather traffic light data. The data is stored and processed in the TLEX I2V Node. From there the data is transmitted to service providers. One of these Service providers will be the interface to the autonomous vehicle. In Figure 47, the dataflow to the autonomous vehicle is shown in red.

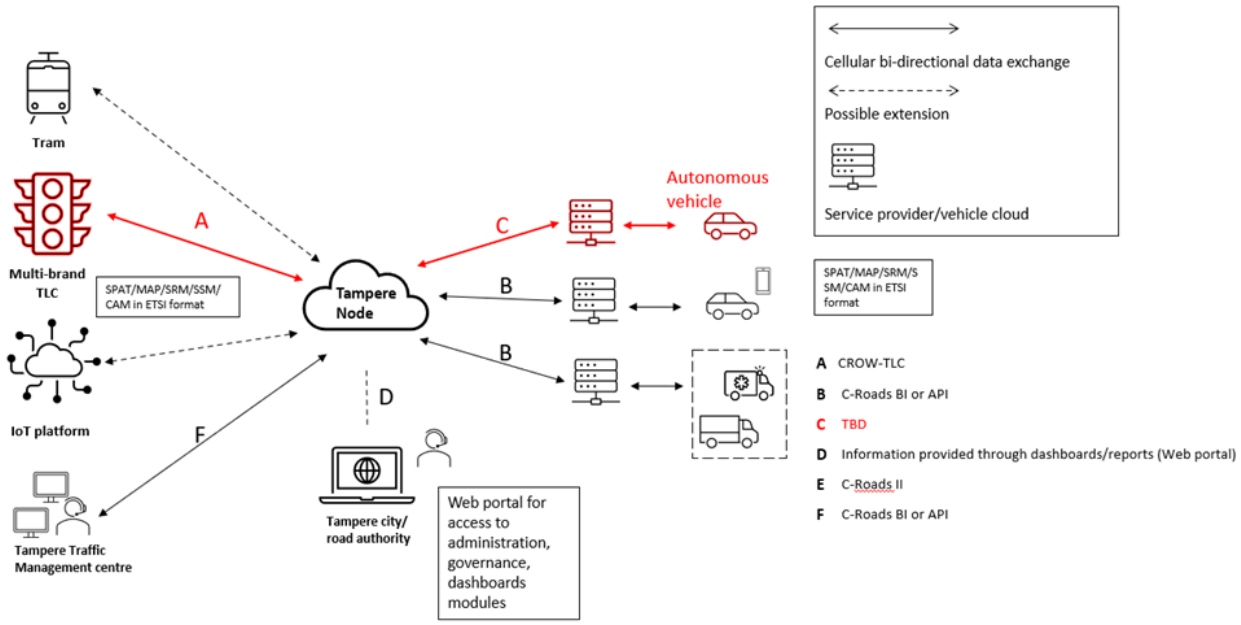

Figure 47: Nodeon Traffic Live Exchange Platform information exchange

Traffic light data is accompanied by timestamps which the vehicle can compare to the current time. This timestamp originates from the device layer, and is provided to the cloud layer where its correctness is checked. If the latency of the data transmitted by the traffic light starts to become too large, the remote control operator of the vehicle will be notified. Then, the remote control operator takes the vehicle over to remote control, i.e. the vehicle, edge tier will be remotely controlled.

From the point of view of the "health dependency":

- If the latency is too high but within safe limits, the system suggests switching from automatic to remote mode (a possible mode change).
- If the latency is outside the safe range, the system will automatically switch to remote mode (a mode change).
- However, if the vehicle has the ability to physically detect the status of traffic lights (e.g., with on-board cameras), it could notify the system of a transition to secondary detection mode and use this information for decision making (a dependency change).

Challenge for application within TRANSACT context

The detailed solution for the technical implementation will be developed in task 3.3. For some aspects this concept is already in-use, mainly in device tier. Nonetheless, the specificity of TRANSACT project and UC1 requires a further elaboration to make the concept fit for a distributed safety-critical cyber-physical solutions.

One of the biggest challenges is to scale this concept to include the other tiers (edge-, cloud- tier), and to adapt it to cross architectural use. Also, in the bigger picture there might be difficulties to always provide secondary (back-up) solutions to ensure an adequate level of safety. This could cause a need for completely different devices, data sources, or systems.

6.7 Concepts for requalification and continuous safety assessment

For safety-critical systems, decision-makers require *safety assurance evidence* that it manages risk acceptably. When developing safety related electronic and programmable control systems, there are a number of sector specific standards and regulations that need to be considered, e.g., EU's Medical Device Regulation (UNION, 5 April 2017), IEC 60601 (IEC, 2005), IEC 82304 (IEC, Health software - Part 1: General requirements for product safety , 2016), IEC 61508 (IEC, IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems, 2010), ISO 26262 (ISO, 2009), and RTCA DO-178B (RTCA, 1992). From a functional safety perspective, the risks are assumed to originate from the product itself, the environment or from the organisations developing the product. Safety measures are therefore defined to avoid and mitigate the effects of potential failures introduced by the electric/electronic and programmable systems. Reducing the risks imposed by intentional misuse or sabotage of a safety related product is, however, out of scope for current functional safety methodology, and existing functional safety standards.

6.7.1 Model-based safety assurance with AMASS

Overview

The certification process depends on the concrete application domain. However, the main ideas are common to all domains. The conceptual basis for certification is that the evidence anticipates the possible circumstances that can arise from the interaction between the system and the environment, to show that these interactions do not pose an unacceptable risk. Although researchers have proposed runtime certification approaches (Rushby, 2008) that could address the challenges of dynamic distributed CPS, none of these approaches are feasible in practice and none of them has been adopted by the certification standards. However, several ARTEMIS and ECSEL projects have addressed safety assurance challenges, related, e.g., to modular, incremental and re-certification (Martin, November 2018), for example, CHESS, CONCERTO, SAFECER, AMASS and SAFECOP. The assurance approach in these projects relies on model driven methodology for the design, verification and implementation of Cyber-Physical Systems, where the components are annotated using assumption-guarantee contracts (Albert Benveniste, 2012) to facilitate the independent development of cooperative safety functions.

Description

In order to assure safety of edge-based CPS, we will extend the open-source toolset from the AMASS project. We will build upon results on contract-based design and related toolset already available in CHESS, SAFECER, CONCERTO and AMASS, which used a modelling language based on an UML/SysML/MARTE profile, offering a component-model contract-based approach, and an open-source tool released under the Eclipse Polarsys ecosystem (Eclipse). This modelling framework also offers rich semantics for modelling extra-functional properties for real-time systems, e.g., related to performance guarantees. The information available in the models and the different verification activities performed on it allows to support the system assurance case

definition, so the argumentation proving that the occurrence of unacceptable risks has been properly mitigated.

The concept is illustrated in Figure 48. The device-edge-cloud systems addressed in TRANSACT consist of several systems where each of those systems has its own accompanying safety case. Each system has a cooperative subsystem co-responsible for the safety function, referred to as a cooperative function in the figure. The left part of the figure (boxes labelled “Hazard and Risk Analysis”, “System and Safety Requirements” and “Independent Assessor”) is concerned with safety assurance analysis. Here we use the System Theoretic Process Analysis (STPA) concept presented in Section 6.5.1.

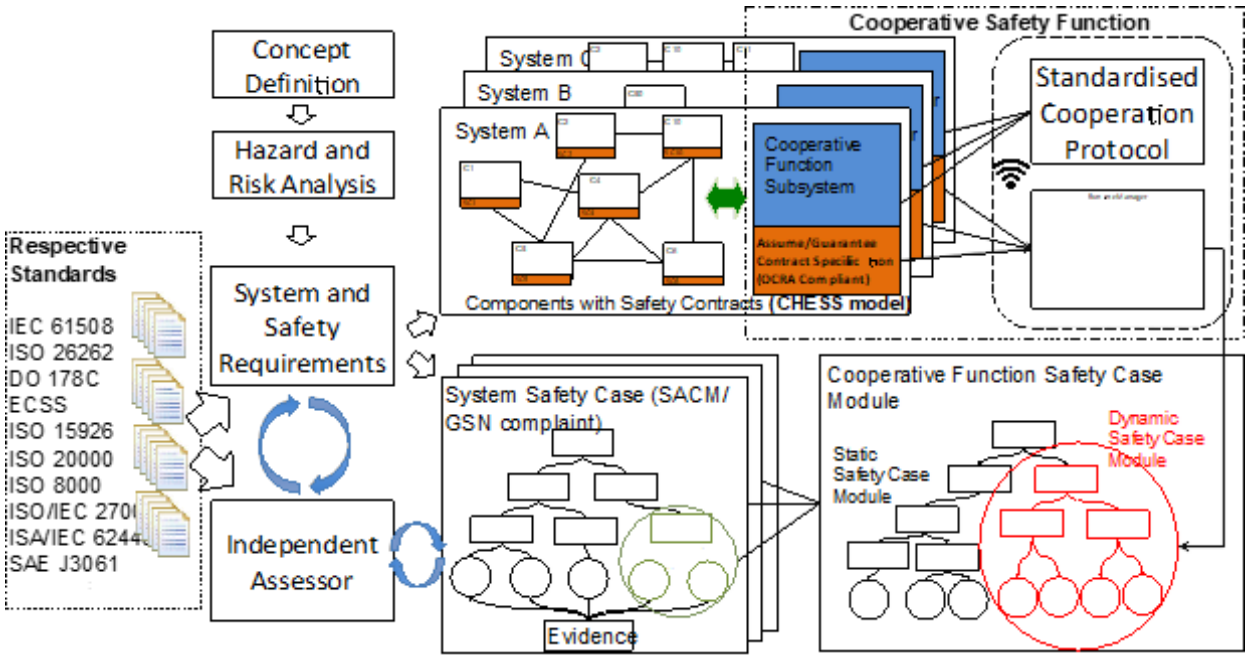

Figure 48 Illustration of the model-based assurance concept

The AMASS project deliverables (AMASS, 2017) give a brief overview of the recommended approach and principles on how to perform a safety assurance analysis of systems-of-systems. It explains a hierarchical approach to ensure that the mission requirements of the system are appropriately allocated by the necessary functional architecture, which in turn is allocated to organizational and technical resources. The safety case relies on constraints that restrict the runtime behaviour of the safety function, in order to anticipate at design time the circumstances that can occur at runtime. These runtime constraints are derived from the safety requirements and are enforced by a safety manager, which includes monitoring of the safety function. We use assumption guarantee contracts to capture these constraints. A contract is a pair of assertions, namely assumptions and guarantees, where an element described by the contract offers guarantees about its own behaviour given that its environment fulfils the assumptions (Benveniste A., November 2012). The safety manager checks such contracts during runtime and generates runtime evidence. The dynamic part of the safety assurance case is built upon the runtime evidence. Since such evidence is tightly coupled with contracts, the resulting safety assurance case depends on the validity of the contract, i.e., both satisfaction of its assumptions and provision of the guarantees when those assumptions are met. We call this a “conditional safety case”: the safety requirements formalized by the contracts are guaranteed only if the related contracts are valid. For example, the integrity of cooperative subsystem A is guaranteed by system A’s *architecture* and safety mechanisms. The public safety evidence of the subsystem B is refined into a set of *demands* that have to be fulfilled by subsystem B in order to ensure its integrity. We assure the satisfaction

of the safety requirements through such contracts and their accompanying runtime evidence by modelling a safety assurance case argument and indicating explicitly which are the dynamic parts of the argument that need to be revisited (the dynamic part of the safety case).

AMASS uses an argumentation editor compliant with the OMG standard SACM (Structured Assurance Case Metamodel) (Group, 2016) for modelling the safety case. The SACM framework supports both main graphical assurance case notations: Goal Structuring Notation (GSN) (Community, 2011), and Claims Arguments Evidence (CAE) (Adelard, 2016). The standardisation and portability offered by SACM enables frequent updates to the safety case with less effort, as it becomes easier to automatically update the dynamic safety assurance case.

Fit with concept TRANSACT reference architecture/components

The safety assurance is performed at the system level. The safety function specification will decide which parts of the device-edge-cloud continuum are involved. In TRANSACT we are interested in safety functions that involve not only the device, but especially also the edge servers, and potentially also the cloud. Therefore, safety assurance touches on all tiers of the architecture, see Figure 49, and it involves the red "Safety-Critical Function" box in the Device and Edge Tiers and the yellow "Mission-Critical Function" box in the Cloud tier.

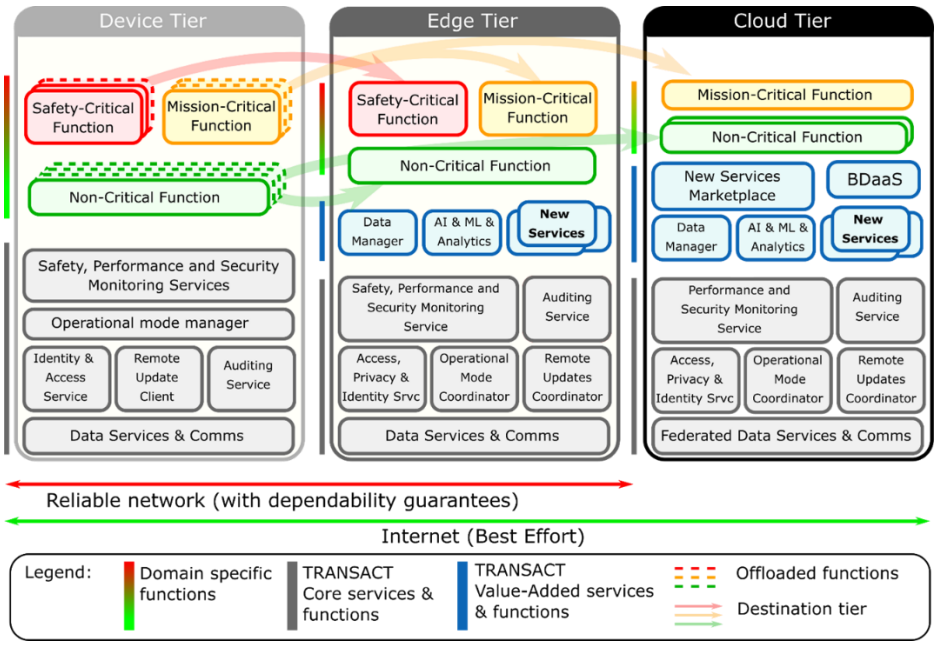

Figure 49 TRANSACT reference architecture

Example in context of Use case 1

The use case UC1 is about the "Remote Operations of Autonomous Vehicles for Navigating in Urban Context". In UC1, the remote operator is responsible for the monitoring and safety in exceptional traffic situations. In case of emergencies, the vehicle is performing a hand-over of control between operator and vehicle is performed in smart way. We will investigate one safety-related function implementing the high-integrity communication between the operator and the vehicle. How to specify and model the communication such that the vehicle can trust a message from the operator, e.g., an emergency brake.

Challenge for application within TRANSACT context

Our solution in TRANSACT is to enrich the AMASS toolset by providing support for development of edge-based systems. More specifically, we will distinguish between the runtime and design time contracts and tightly couple the contracts with the corresponding safety assurance evidence to address the dynamic safety assurance nature of edge-based CPS. At design time, the architecture design, demand-guarantee specification, and safety-case composition mechanisms are verified. At runtime, a safety assurance manager performs checks of demand and guarantee and composes the safety cases; also, a monitoring module performs data acquisition to collect safety-related data, which is used to periodically recheck the evidence related to the safety cases.

Safety assurance will check that runtime management of safety and security is sufficiently integrated in the safety-critical distributed system. Runtime monitoring will provide sufficient information for diagnosis of the components. Fall-back mechanisms are built in the system that mitigate faults and attacks. Update mechanisms will prevent unauthenticated use and ensure continuous functioning of the distributed safety-critical CPS solution. The results will provide a basis for extending existing standards and recommendations as well as contribute to future guidelines and rules for certification of edge-based distributed safety-critical service solutions.

We will demonstrate the extended AMASS methods, tools and techniques for the analysis and validation of the performance metrics of distributed safety-critical CPS. These are based on monitoring, design-time and runtime (re)configuration of edge platform resources, offering both performance guarantees for hard real-time applications and graceful degradation for soft real-time applications.

6.7.2 Continuous safety assessment within DevOps

Overview

Satisfying regulatory safety requirements is a big challenge in context of safety-critical systems. Nowadays, in ever-increasing demand for innovation speed, a need to adapt current development and operations pipeline to include safety assessment is of crucial importance. This would enable the continuous delivery of software in safety-critical systems and would be corner stone in automation of the safety assessment process in the delivery pipeline (Zeller, 2021). In what follows we introduce the concept of a delivery pipeline in context of DevOps which is adopted from (Zeller, 2021)[see Figure 50).

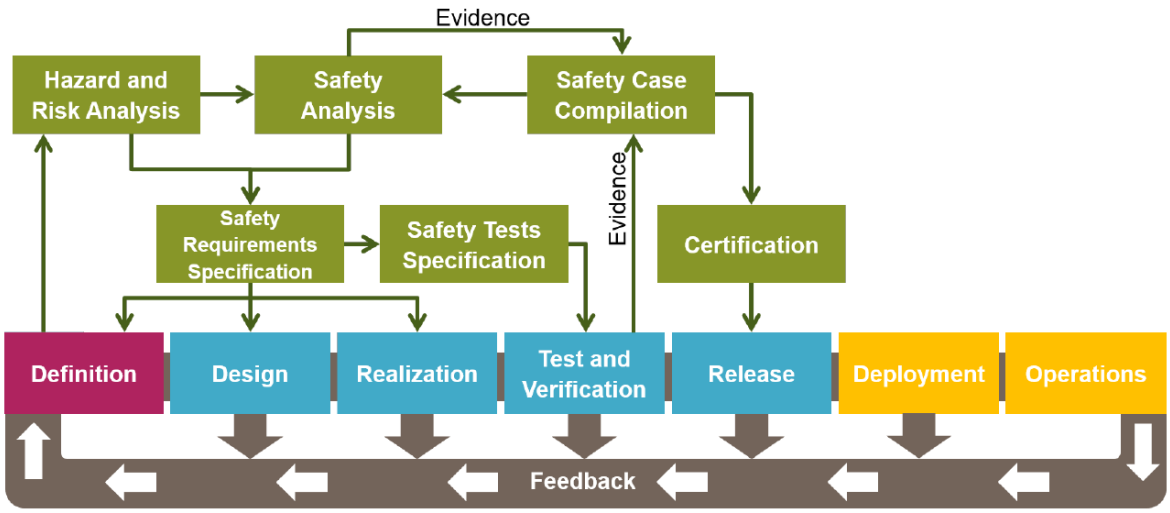

Figure 50: Continuous delivery pipeline with additions for safety-critical CPS development (Zeller, 2021)

Description

Depicted in Figure 50 is a high-level characterization of embedding safety in the continuous delivery pipeline. In bottom a conventional DevOps loop is depicted while the green boxes show the breakdown of safety pipeline and their contribution to the overall DevOps process.

In the following a brief description of different sections of the pipeline is presented.

DevOps pipeline

In DevOps traditional software engineering roles in which development was separated from operations are merged to improve communication and enhance the production release frequency. In bottom of the Figure 50 an illustration of the main component of workflow in a DevOps approach is shown. As can be seen the feedback/monitoring is continuously drawn from different steps to improve the communication while allowing a more synergic release cycle.

Safety assessment breakdown and its connection to DevOps pipeline

Safety is a system quality that emerges through the life cycle of system. As illustrated, in the context of DevOps the enhancement of release frequency is of crucial importance and therefore, in order to obtain a seamless release cycle for safety critical Cyber-Physical Systems we need to merge the safety assessment in different steps of the DevOps pipeline. The Upper part of Figure 50 reflects a breakdown of the safety assessments and their relation to the DevOps pipeline as presented in (Zeller, 2021) based on required steps in safety standards such as IEC61508.

Furthermore, to increase the speed of releasing, it is crucial to automate the safety assessment pipeline through adaptation of model-based techniques. An example of such an adaptation is provided in Figure 51. This is in line with the line of practice to integrate safety in model-based system engineering approaches (Ahlbrecht & Durak, 2021), (Albrecht & Bertram, 2021).

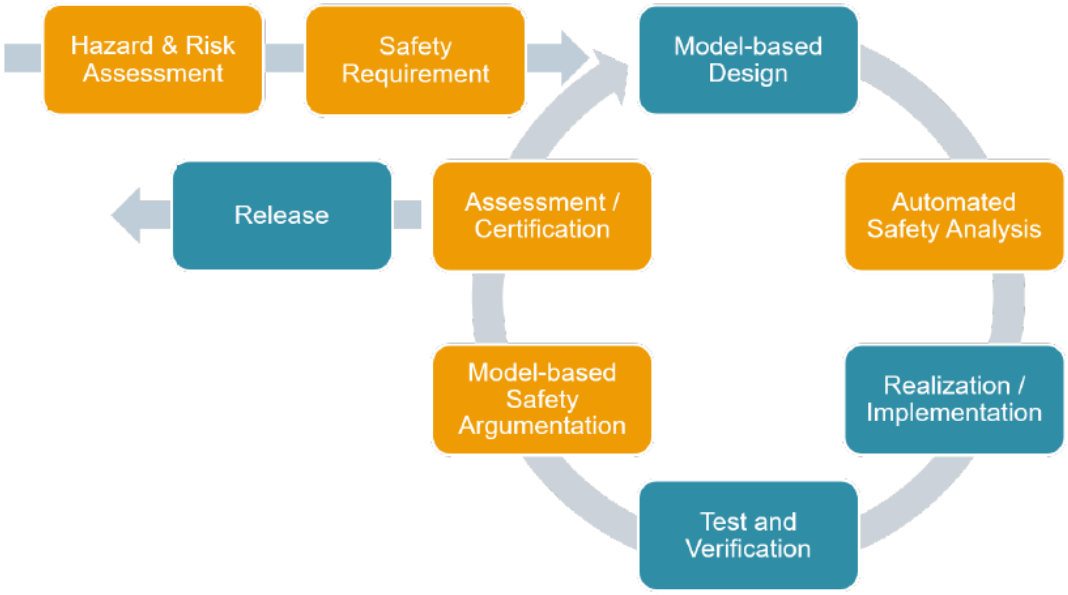

Figure 51: Continuous safety assessment pipeline using model based-techniques (Zeller, 2021)

Fit with concept TRANSACT reference architecture/components

Although DevOps can be applied to monolithic software systems, its true power manifests in the concept of microservices which are widely used in cloud-based systems (Balalaie, Heydarnoori, & Jamshidi, 2016). Furthermore, TRANSACT has a strong emphasis on the implementation of safety critical systems in cloud-edge-device continuum which bring the question of how to asses safety while benefiting speed of innovations in context of cloud-edge-device applications. Therefore, it is of high importance that a pipeline for automated safety assessment that covers the applications in device-edge-cloud continuum is developed within the TRANSACT project (see Figure 52).

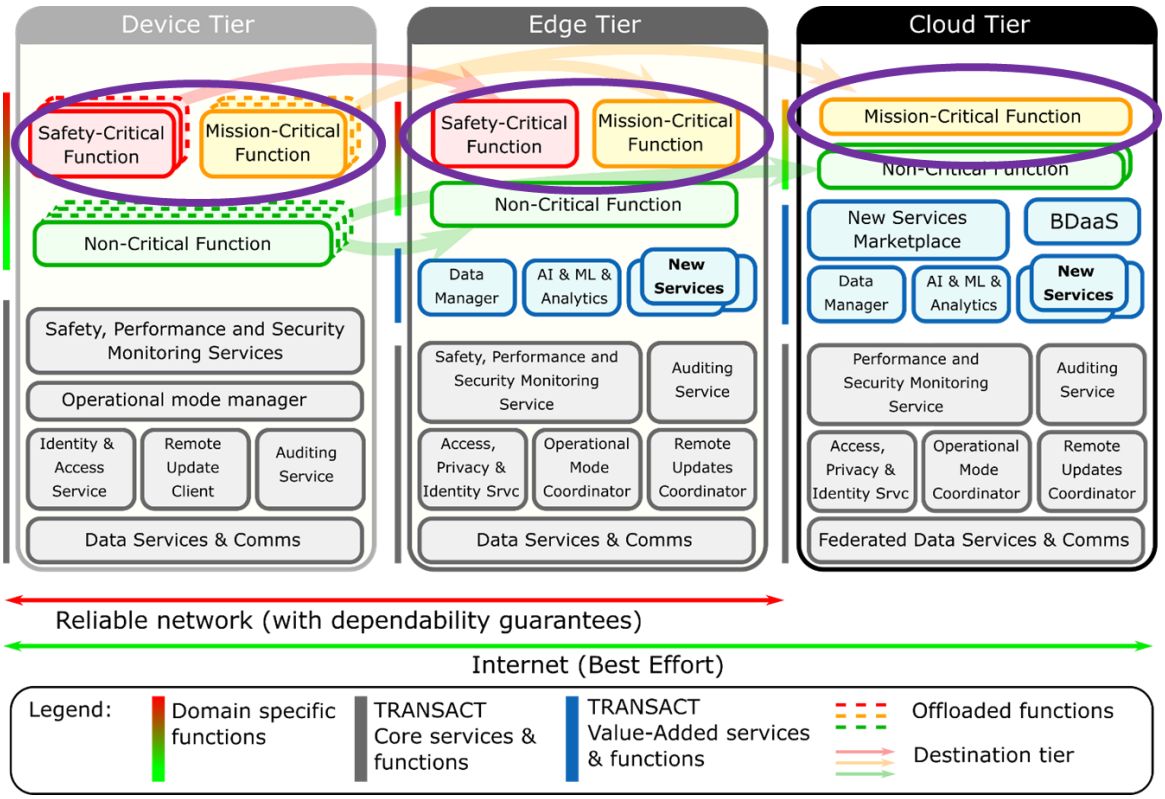

Figure 52: DevOps safety assessment impact on critical functions TRANSACT reference architecture

Example in context of Use case 4 (model registration service)

Incremental updates of a model registration service (see Section 6.5.1, example in context of a use case), will need to be assessed before deployment and use. Parts of the STPA safety analysis (Section 6.5.1) may be automated and made suitable for the continuous safety assessment pipeline (see Figure 51) in order to benefit from the speed of innovations in possible in context of cloud-edge-device applications

Challenge for application within TRANSACT context

Application of continuous safety assessment pipeline in context of a device-edge-cloud continuum CPS is quite novel. In TRANSACT experiments with STPA (Section 6.5.1) and AMASS (Section 6.7.1) are planned to investigate the steps needed towards realising a continuous safety assessment pipeline.

7 Platform concepts for safety and performance

7.1 Introduction

This section describes platform concepts.

Platform is the environment in which the application is executed. It comprises of the complete infrastructure in the device, edge, cloud continuum to execute the application, including hardware, operating systems, hypervisors, communication networks, containers and cloud computing services .

7.2 TRANSACT project harmonised needs and expectations

As part of the TRANSACT project and following the TRANSACT reference architecture, use case applications will be divided over the device-edge-cloud continuum, including safety-critical and mission-critical functions. The distributed platform on which these functions are deployed, needs to have specific capabilities, in order to support this distribution. Furthermore, as a multitude of devices could potentially make use of the same edge or cloud infrastructure, demand may vary as function of the number of connected devices and service requests.

Platforms hence need to offer predictable and scalable configuration options to support active online management and scaling of heterogeneous resources without jeopardizing performance and SLA of active applications and services. To enable this, specifications for platform services and service levels must be made accessible to on-line management strategies. Only proper usage of a well configured platform's resources and assets can ensure that the distributed device edge-cloud CPS fulfils safety, performance, security, and privacy requirements. When safety-critical or mission-critical functionality is offloaded to the edge or cloud, then this requires special means and measures to make these parts of the distributed solution (and their interlinked communication and network) fit to support such safety-critical or mission-critical functionality.

The complete set of technical requirements is documented in TRANSACT deliverable D1.2. This section describes selected platform concepts that address these requirements, and lists open challenges for these concepts as applicable that will be investigated in the course of the TRANSACT project.

7.3 Selected platform concepts for safety and performance

The selected platform concepts are highlighted in Figure 53.

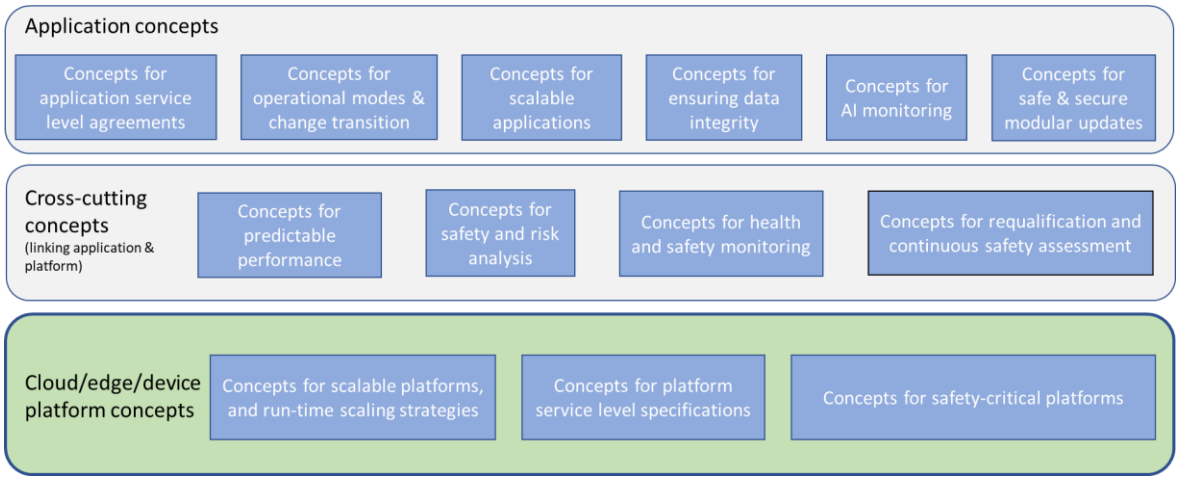

Figure 53: Selected platform concepts

7.4 Concepts for scalable platforms, and run-time scaling strategies

Cloud platforms provide efficient infrastructure that helps to build high-performing and scalable distributed applications and provide tools to dynamically adjust the applications' resources to accommodate high or low demanding workflows. However, only proper usage of provided platform's resources, services, and tools can help to design and build the end system that fulfils the needed scalability and performance requirements.

The following sections elaborates about cloud platform offering in terms: 1) what hosting environments are available to build scalable solutions; 2) what are the resource types that can be used to build scalable solutions; and 3) what are the means to ensure dynamic scaling of the applications.

Figure 54 highlights the impact of the platform scalability aspects on the core TRANSACT reference architecture components.

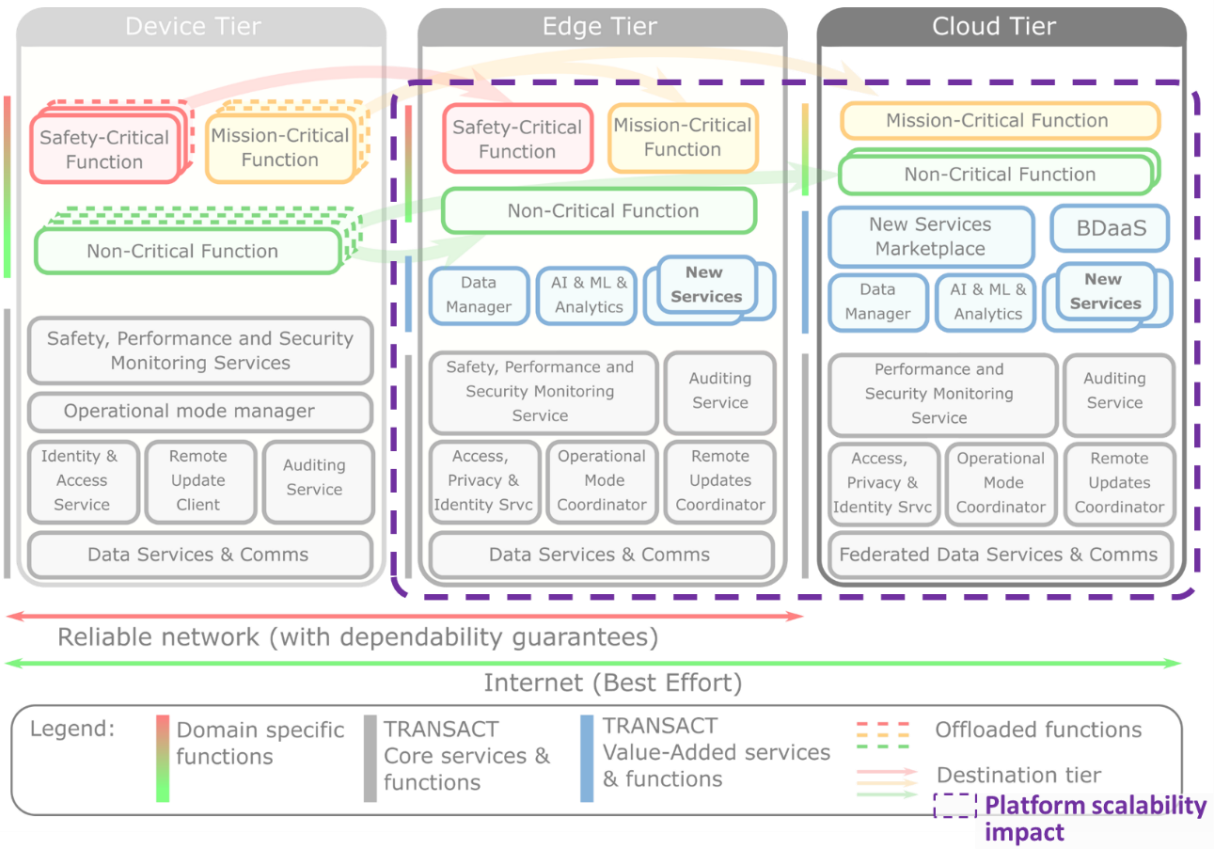

Figure 54: TRANSACT reference architecture with impact of platform scalability

7.4.1 Concepts for hosting/virtualization: VMs/Containers/Serverless computing

The cloud platforms support wide range of workflows to accommodate various applications designs, usage patterns, and configurations. Therefore, the cloud platforms allow to build solutions that may be deployed using various computation assets. Typically, the cloud platforms allow deploying using virtual machines (VM), containers, or serverless computing solutions.

7.4.1.1 Virtual Machines (VMs)

A virtual machine (VM) provides virtualization/emulation of a computer system that function as a virtual computer system with its own CPU, memory, network, and disk storage. The cloud platforms provide ready to use pre-configured VMs in different families and sizes that offer different compute (CPU/GPU/FGPU/…), memory, storage (SSDs/HDD/…), and networking capabilities to accommodate for variety of workflows (see Section 7.5). Those capabilities can be controlled using API which allows experimentation with the available resources to achieve optimal performance and cost aspects per required workflow.

The move from physical HW to VMs deployed in the cloud can be easily utilized even by the existing applications with minimal changes effectively supporting the “Lift&Shift” scenario.

7.4.1.2 Containers

A container provides virtualization of the operating system that allows full isolation of the application deployed within the container. Specifically containers provide 1) portability (a container image is a standalone package with all needed binary dependencies and configurations which makes them easy to deploy), 2) are lightweight compared to virtual machines (they share OS resources that makes it possible to pack multiple containers onto a single node, especially, when the application consists of many small services); 3) resource isolation (the amount of memory and CPU can be configured per container ensuring proper resource distribution to the relevant parts of the application services).

Typically, the cloud platform provides services 1) to manage service orchestration (deployment service health monitoring , configuration updates), 2) to load balance network traffic across the service instances, 3) service discovery, and 4) scale containerized applications automatically. Depending on the application architecture, the containers can be deployed on dedicated platforms or on the provisioned virtual machines.

The microservices architecture style applications and solutions can take advantage of container-based deployment.

7.4.1.3 Serverless computing

Serverless computing abstracts the execution environment from the application code, so, the process of deploying the application is greatly simplified. Specifically, the serverless applications are not concerned with any deployment aspects related to containers, VMs, or physical servers, such as capacity planning, configuration, management, maintenance, fault tolerance, or scaling of underlying infrastructure. In short, serverless computing enables running any type of application code with minimal deployments concerns.

Serverless computing is very cost-effective solution as it uses resources only when the application code is running/executing. The execution might be triggered manually, or as a result of a timed process, an HTTP request, or a file upload. The serverless-based applications can work in conjunction with components deployed in containers or virtual machines.

Highly modular and stateless application design can take advantage of serverless-based deployment.

7.4.1.4 Concept summary and impact within TRANSACT

The summary of Hosting/Virtualization concepts is presented in Table 4.

Table 4: Summary of Hosting/Virtualization concepts: VMs/Containers/Serverless

	VM	CONTAINER	SERVERLESS
Application scale unit	VM	Image	Code function
Abstracts	HW	OS	Runtime
Lifespan of the instance	Days-Months	Minutes-Days	Msec.-Minutes
Responsibility to build and deploy the application	OS, runtime, application, application's dependencies	runtime, application, application's dependencies	Function deployment

Example in context of a use case

Every use-case needs to be executed with required performance and reliability in a safe way. Therefore, depending on the use-case needs the system architecture may use any of the hosting/virtualization solution to ensure fulfilling the requirements. For example, the AI algorithms for data analysis may use serverless solution whereas the legacy applications (e.g., healthcare viewing application) may utilize virtual machines to take immediate advantage of the new cloud/edge deployment environment.

Challenge for application within TRANSACT context

The usage of the hosting/virtualization concepts very much depends on the system/application architecture and its building blocks. Therefore, the main challenge for the applications is to investigate which concepts can be used directly for the current components or the system architecture/design needs alteration. For example, the monolithic-based architecture systems may not be able to use containers effectively and cannot use serverless solution at all so they are constrained to the virtual machines. For service-based architecture both the virtual machine and contains could be a valid option. Another aspect is the cost of the solution deployed in the edge/cloud, i.e., virtual machines costs are higher as they are charged for running VMs (despite if VM is doing anything useful or not), in contrast, in the serverless solution the cost are only charged when running (using resources).

7.4.2 Resource types

The cloud platforms need to support a broad range of use-cases with a wide set of non-functional requirements. Therefore, in order to ensure optimal fulfilment of those requirements (especially in terms of performance and reliability) the system architecture may need to choose different solutions depending on the required capabilities delivered by the components within the system.

The cloud platforms allow provisioning different types of cloud resources to optimize fulfilment of the requirements and balancing the costs at the same time. Typically, there are generic resources to choose from that can be fully optimized but there are also preconfigured resources focused on computing and data storage workflows with appropriate networking capabilities. Next to the generic resources, there are also specialized resources configurations optimized for the specialized workflows, such as: high-performance computing (HPC) workflows, AI and machine learning workflows, big data analytics workflows, etc.

Often, multiple approaches are required to get optimal performance across a workload, therefore, well-architected systems use multiple solutions and enable different features to improve performance.

7.4.2.1 Computing resources

There are many different computing-bound workflows depending on the application design or usage patterns. So, there is a need for different configurations of the compute resources in order to optimally address the application's needs. Table 5 summarizes what typical computing resources configurations are provided by the cloud platforms.

Table 5: Typical computing resources configurations provided by the cloud platforms

RESOURCE TYPE	DESCRIPTION
General purpose	Typically, VMs with balanced CPU/memory/storage/networking ratio that are ideal for generic workflows with small to medium databases and low to medium traffic web services. This class of resources usually is available in many (balanced) configurations varied by the vCPU and memory chips models, speed and size of available local storage, or network bandwidth limits.
CPU optimized resources	Typically, VM ideal for compute bound applications that benefit from high performance processors, such as batch processing workloads, high performance applications and web services. This class of resources usually is available in many configurations where main focus is put to ensure high vCPU and memory data bandwidth with limited storage and networking capabilities.
GPU optimized resources	Typically, VM ideal for applications requiring intensive computation that can be done more efficiently on GPUs than on CPUs, such as graphics-intensive workflows, floating point intensive calculations, visualization or data pattern matching processing. This class of resources usually is available in many configurations where main focus is put to availability of GPUs and memory resources with matching capabilities of vCPU, storage and networking resources.
Memory optimized resources	Typically, VM ideal for memory bound applications that require processing large data in memory, such as medium to large caches, in-memory analytics, or database servers. This class of resources usually is available in many configurations where main focus is put to ensure large size memories able to hold many data with secondary focus on the vCPU capabilities and with limited storage and networking capabilities.
Disk optimized resources	Typically, VM ideal for the applications requiring high disk throughput and I/O on local storage, such as data warehousing, large (SQL/NoSQL) databases, or quick access to the data on local storage such as search engines. This class of resources usually is available in many configurations where focus is put to ensure effective and performant storage with effective vCPU and networking capabilities.
Specialized resources	The specialized solutions are architected for optimal resource usage. Such dynamic scaling allows customers to focus on defining the requirements for the workflow without concerns about the needed infrastructure. The specialized solution optimizes usage of compute, storage, and networking resources for addressing the workflow optimally in terms of application needs cost-effectively. An example of such solution could be dedicated high-performance computing (HPC) resources designed to help solving very intensive and demanding workloads such as: genomics, oil and gas simulations, finance, weather modelling, etc.

7.4.2.2 Data storage resources

Data storage resources help to optimize the storage needs of the solution, especially, when multiple different storage aspects are required to guarantee the system performance and reliability. The cloud data storage resources are available in many types depending on the needed database support (SQL, NoSQL, …), the access method (block, file, or object), access pattern (random or sequential), throughput required, access frequency (online, offline, archival), availability and durability constraints.

7.4.2.3 Network resources

Since the network connects most of the application components/services, it can have positive and negative impacts on solution performance and reliability. Especially for the solution based on the distributed environments (both publicly accessible and private) the network considerations play important role to address possible interconnectivity challenges (failures, latency, etc.) but also to accommodate solution growth and integration with other systems.

Network resources help to optimize and improve reliability of a workloads that require controlled latency, throughput, jitter, bandwidth, or security aspects. The cloud platform can offer different network solutions to accommodate those needs such as selected communication protocols (e.g., TCP vs UDP), network encryption, dedicated network and their redundancy, utilization of regions and availability zones (see Section 7.5.1), etc.

7.4.2.4 Concept impact within TRANSACT

Example in context of a use case

Most of the TRANSACT use-cases depends on AI services. Therefore, ensuring that those AI data intensive algorithms perform optimally they may be deployed on highly optimized CPU hosts with high speed disk access. Another example, is the image-processing intensive applications (like in the healthcare use-cases) that may benefit from running with GPU optimized resources to achieve the desired performance.

Challenge for application within TRANSACT context

The usage of various resources depends very much on the use-cases and the system/application architecture. Therefore, the main challenge for the TRANSACT applications and services is to investigate which workflows require what resources and if the architecture and design can effectively utilize those resources.

7.4.3 Cloud platform scaling and load balancing

Cloud platforms support elastic scaling of resources by providing a seemingly endless pool of resources, i.e., the end system can be configured to dynamically increase the amount of compute resources during peak loads and return to normal automatically when the peak drops. This allows to react in real time to abnormal condition in order to guarantee needed system performance and reliability. Scalability provided by the cloud platforms is possible due to virtualization of the compute, network, and storage resources allowing adaptable provisioning of those resources.

Two main approaches in which cloud platforms support scaling are: scaling up (vertical scaling) and scaling out (horizontal scaling):

- **Scaling up** is based on increasing capacity of a resource (e.g., by using a more powerful virtual machine in term of compute, storage, or networking, see also Section 5.6 for possible choices that can be made to ensure optimal resources availability), whereas,
- **Scaling out** adds new instances of a resource (e.g., adding more virtual machines of the same capabilities, adding more database replicas, or more services of a given type).

An advantage of scaling up is that it can be used without making major or any changes to the application as demands increase. However, the scaling up has its limits where it cannot be applied anymore. At that point, any further scaling must be horizontal (scaling out). The scaling out has better fit into the cloud elastic capabilities, can be more cost effective (e.g., running a number of smaller VMs may be cheaper than using a big VM), and can improve system resiliency, by adding in easy way redundancy. However, scaling out can be effective only if it is designed into the system, i.e., it has to be based on loosely coupled, stateless components/services/functions that can be added or removed at any time without impacting the system

integrity. The load balancing is a standard mechanism provided by cloud platforms to balance a workload across resources in scaling-out scenario.

The cloud platforms provide **autoscaling** that enables to run the right number of resources to handle the current load on the system, i.e., it adds resources to ensure performance and reliability during increased load (such as seasonal workloads), but it removes idle resources to save money during decreased load (such as nights or weekends). Scaling can be performed on a schedule, or based on runtime metrics, such as CPU or memory usage (for example, if average CPU usage in an instance is above 70% then a new instance is added, or if CPU usage falls below 50% then one instance is removed). An example of autoscaling is presented in Figure 55.

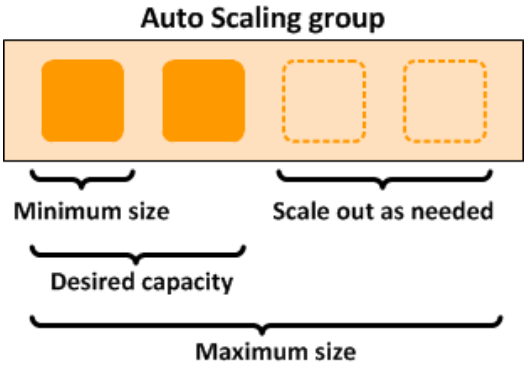

Figure 55: Amazon AWS EC2 Auto Scaling explained (AWS-scaling, 2022)

Usually, the managed cloud platform's resources and services allow scaling out and have autoscaling built in. However, to ensure effective and reliable usage of resources the autoscaling is supported by the resources load balancing. The **load balancing** mechanism ensures the even load distribution (of incoming application traffic) across a group of backend resources or servers. This way load balancing increases system availability by distributing resources within and across (availability) zones and ensures that no instance is overwhelmed.

Figure 56 shows the Amazon EC2 auto scaling and load balancing in action with Availability Zones, i.e., the Amazon EC2 auto scaling attempts to launch new instances in the Availability Zone with the fewest instances. However, if the attempt fails, then the Amazon EC2 auto scaling attempts to launch the instances in another Availability Zone until it succeeds. The load balancing ensures that all available instances are used evenly.

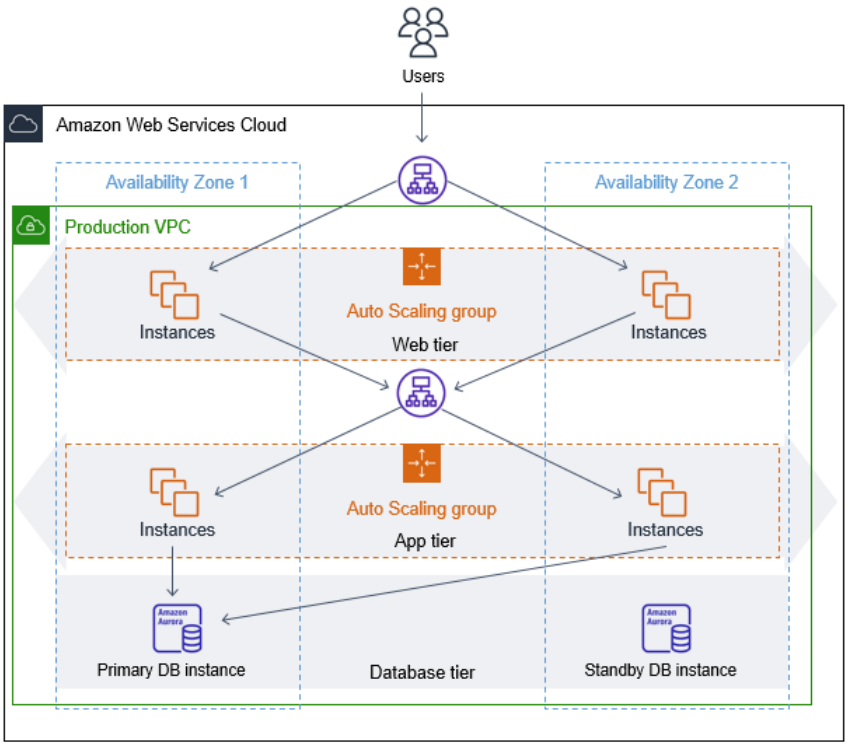

Figure 56: Amazon AWS EC2 Auto Scaling with load balancing example (AWS-scaling-and-balancing, 2022)

Example in context of a use case

Every use-case needs to be executed with required performance and reliability. Therefore, depending on the use-case appropriate number of the services should be available to ensure fulfilling the requirements. For example, having multiple instances of AI-based services processing healthcare data coming from various hospitals may require scaling-out during the regular hospital working hours and scaling-down during the night where there is less demands on data processing.

Challenge for application within TRANSACT context

The usage of (auto) scalability and load balancing depends very much on the use-cases and the system/application architecture. Therefore, the main challenge for the applications is to investigate if the application/system architecture and design is based on the independent building blocks that would allow to effectively utilize scalability/load balancing concepts with acceptable costs.

7.5 Concepts for platform service level specifications

Cloud platforms provide efficient and reliable infrastructure that helps to build high-performing, reliable, and secure distributed applications and provide tools to collect meaningful metrics to verify the applications behaviour in terms of functionality and used resources. However, only proper usage of provided platform's resources, services, tools and assets guarantees can help to design and build the end system that fulfils the needed safety, reliability, performance, and security requirements.

The following sections detail 1) concepts for high availability provided by the cloud platforms and their services to be leveraged in the solution design; 2) flexible deployments with hybrids clouds solutions, and 3) what telemetry data can be collected from the platform to validate the expected service level specification and how to enhance the system monitoring with relevant metrics to effectively diagnose system failures.

Figure 57 highlights the impact of the platform tools and solutions on the core TRANSACT reference architecture components.

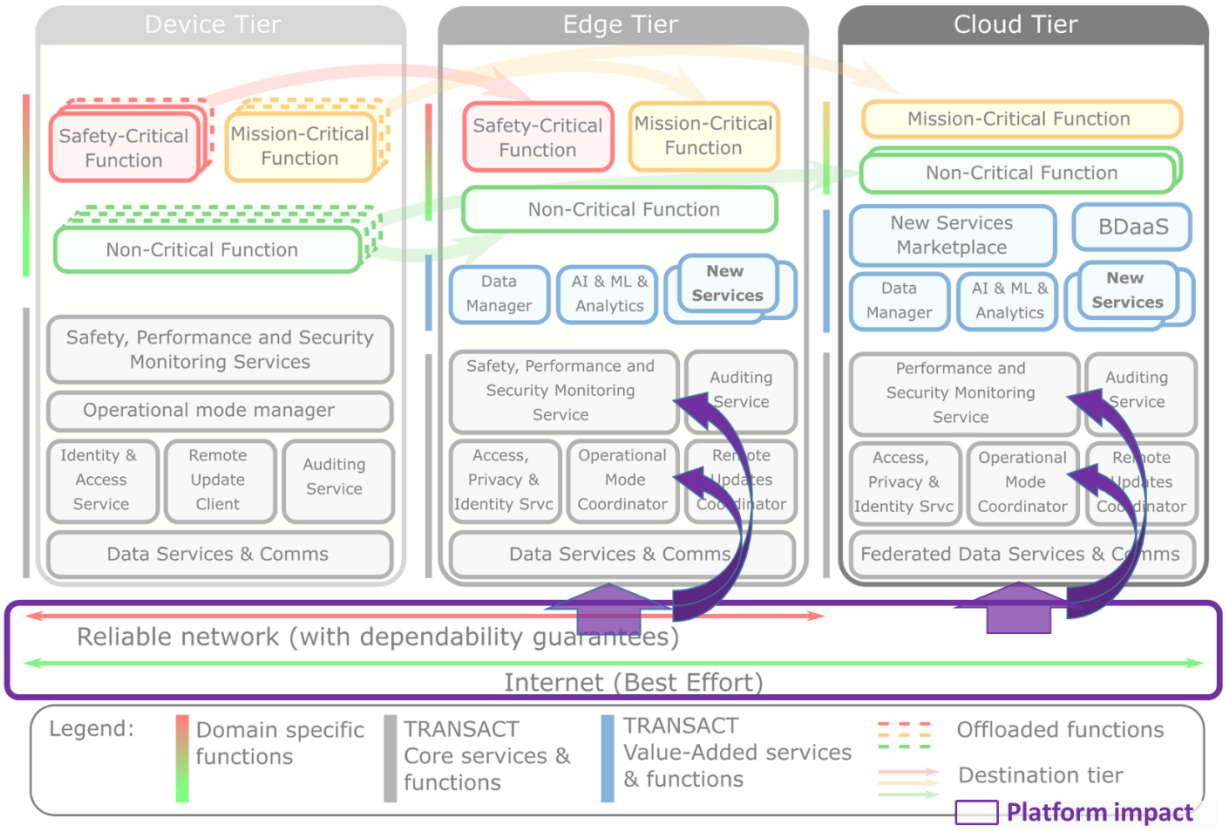

Figure 57: TRANSACT reference architecture with impact of platform service levels

7.5.1 Regions and availability zones for high platform resources availability

In general, the cloud platforms infrastructure is engineered to resist failure. The design of the infrastructure utilizes the notation of the fault domains that limit the effect of a system failure to subset of applications or services, so the resources outside of the fault boundary are unaffected by the failure. Using multiple fault isolated boundaries helps to limit the impact on overall system availability.

The *regions* and *availability zones* are the core concepts of the cloud platform infrastructure that can be used to build high resiliency and reliable end solution. *Regions* are a collection of data centers connected through a dedicated regional low-latency network to ensure high performance and security. *Availability zones* are physically separated locations within each region that are tolerant to local software and hardware failures due to events such as earthquakes, floods, and fires. To ensure resiliency, there are at least two separate availability zones within the region that are connected by a high-performance network with a predefined round-trip latency (usually around 2ms) to ensure quick data synchronization and accessibility in case of failure (Figure 58 shows example setup of regions and availability zones in Microsoft Azure cloud platform). Each availability zone is composed of data centers equipped with independent power, cooling, and networking infrastructure. Typically, the cloud platform's services are zones-enabled which allows to configure them for auto-replication (e.g., based on a periodic schedule or changes in the datasets) across availability zones. Typically, the load balancing and auto scaling services help distribute load across resources deployed in various availability zones, so in case of failure, the traffic towards resources in unhealthy zones

is routed to the healthy ones. This allows to design highly available system when the services are automatically transition between zones without interruption.

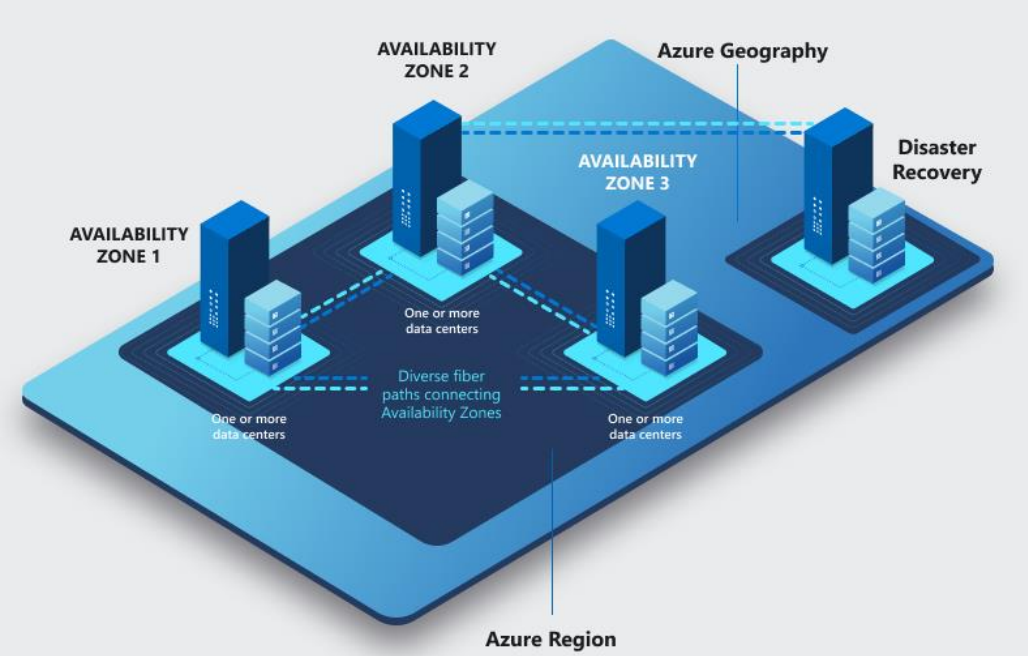

Figure 58: Microsoft Azure regions and availability zones example (Azure-availability-zones, 2022)

The single region with multiple availability zones provides very high reliability and protects against large local disasters. However, to further improve system reliability and ensure continuation of critical workflows it is possible to setup *cross-region* disaster recovery by replication of needed resources. Multi-regional solutions improve further the system availability by introducing physical isolation of data centers which reduces the likelihood of natural disaster, civil unrest, power outages, or physical network outages.

The high availability of the cloud platform infrastructure build on the geographical distribution across (multi-)regions and availability zones, has foundation in high redundancy of hardware resources (such as hard drives, power supplies) and round-the-clock monitoring and part replacements to ensure as far as possible services availability.

The cloud provided services guarantees for performance and reliability are strongly coupled to the chosen design and services deployment models within the availability zones and the regions.

Example in context of a use case

Depending on the use-case an appropriate number of the services should be available to ensure fulfilling the requirements. For example, having multiple instances of AI services (deployed in multiple availability zones) processing healthcare data coming from various hospitals may require high availability to ensure that diagnostic information (extracted from the processed image) is provided always and in time so the doctor gets the information that helps in providing better patient diagnostics.

Challenge for application within TRANSACT context

The high system availability depends very much on the use-cases need and the system/application architecture. Therefore, the main challenge for the applications is to investigate which parts of the system that realize the critical use-cases require high availability components that may impact system safety,

performance, or reliability. Another, challenge is to ensure by design that in case of failure the recovery of the system functionality (switch over to the healthy instance) has minimal impact on the critical system functionality.

7.5.2 Concept for flexible deployment: hybrid clouds

Hybrid clouds solutions allow deploying the services across the edge/cloud continuum depending on the needs of the system applications. This cloud platform capabilities gives flexibility needed for applications with low-latency requirements, safety constrains, or data residency constraints. The hybrid clouds can also be deployed on the infrastructure already available on the customer premises. Figure 59 shows an example of, Google Anthos (Google, 2022) offering.

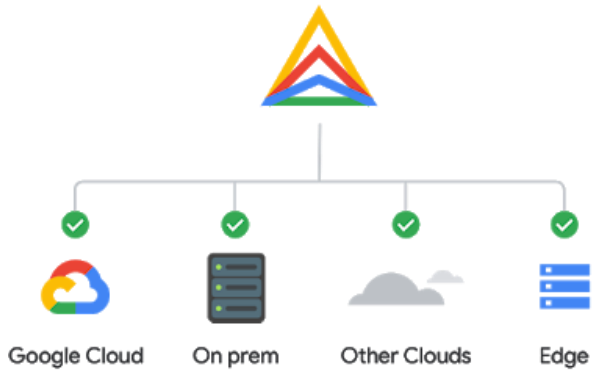

Figure 59: Google Anthos example of hybrid cloud solution (Google, 2022)

Example in context of a use case

Depending on the deployment environment constraints the system services can be deployed in the cloud or on premises. For example, in case of healthcare use-case the system deployment very much depends on the hospital infrastructure therefore for poor-network connectivity infrastructure it may be better to deploy the services on-prem whereas, with efficient network connectivity infrastructure hospitals the service may be deployed—in both scenarios the same applications and services can be deployed without changes. Another example could be data availability that may be constrained to the hospital premises and should not be stored in the cloud—in this case, still the system may be deployed without change taking advantage of hybrid solution model.

Challenge for application within TRANSACT context

The hybrid clouds solution supports the edge/cloud continuum by easing the seamless application deployment on the edge and on the cloud. Therefore, the main challenge for the TRANSACT applications is deciding where the applications should run effectively for a given workflow in a specific environment and how to ensure cost effective configuration of such deployment (e.g., sometimes services can run effectively from within the cloud but there could be environments with (occasional) poor connectivity that would require on-premise deployment).

7.5.3 Concepts for platform resources monitoring (Metrics)

Cloud platform support extensive monitoring and health management of the infrastructure (where the system applications and services are deployed) and the platform's services the deployed solution depends on. Next to the raw metrics the cloud platform provides related visualization and analytics tooling that can

consume and visualize the metrics. In addition, the metrics can be accessed also via API for easy integration with other services or monitoring systems.

Specifically there are dedicated monitoring capabilities and accompanied cloud platform tools ((AWS-monitoring, 2022), (Azure-monitoring, 2022)) such as:

- **Compute tier monitoring**: exposes metrics from compute resources on which the application and services are running, such as virtual machines, containers, functions. Example metrics are: CPU usage, memory utilization, I/O rates, runtime errors (e.g., failure to open a file).
- **Data tier monitoring**: exposes metrics about the databases, storage accounts, and other data sources that interact with the application. Example metrics are: IO read/write times, disks errors, data volumes.
- **Network tier monitoring**: exposes metrics related to the network traffic and tools for easy network troubleshooting of connectivity problems. Example metrics are: throughput, error rates, request/response times, bandwidth consumption

To ensure overall system health and its performance, availability, security, and reliability the cloud platforms provide alerting capabilities that can use the monitoring (and other instrumentation data) to generate notifications when a significant event is detected. Effective alerts help quickly get started on system diagnostic but can be also used for effective system scaling (due to resource/service health issues) or enable cost-awareness by alerting about exceeding the budgets and limits.

7.5.3.1 Platform telemetry/monitoring API

Instrumentation is a critical part of the monitoring process because only if relevant data is collected then the meaningful decisions can be taken about the performance and health of a system. Typically, the cloud platforms provide resource monitoring APIs and central logging systems for integration within the applications to ease the realisation of effective system diagnostics functionality.

7.5.3.2 Concept impact within TRANSACT

Example in context of a use case

In each TRANSACT use-case the telemetry information would help to check the use-case health and its derivate statistical information about its performance. For example, in case of healthcare use-case the telemetry information can provide insights about average length of the performed health procedure, how long does the image processing takes, which systems functions are used most often so they can be optimized for the doctor usability and efficiency.

Challenge for application within TRANSACT context

The main challenge for the applications is to take full advantage of the platform telemetry and monitoring capabilities in order to improve its efficiency and reliability. On top of the available platform telemetry information, the applications should design and implement their own telemetry such that it can be easily integrated with the platform telemetry in order to learn about whole system health and drive the improvements where applicable.

7.6 Concepts for safety-critical platforms

Safety-critical systems are systems whose failure or malfunction may result in one (or more) of the following outcomes: death or serious injury to people, loss or severe damage to equipment/property, environmental harm. Hence, safety-critical systems need support and safeguards in the platforms to ensure that they do not fail or malfunction.

Depending on the level of safety requirements, several operation regimes may apply as follows (Wikipedia, 2021):

- **Fail-operational systems** continue to operate when their control systems fail. Examples of these include elevators, gas thermostats in most home furnaces, and passively safe nuclear reactors. However, fail-operational mode is sometimes unsafe. For example, nuclear weapons launch-on-loss-of-communications was rejected as a control system for the U.S. nuclear forces because it is fail-operational: a loss of communications would cause launch, so this mode of operation was considered too risky.
- **Fail-soft systems** are able to continue operating on an interim basis with reduced efficiency in case of failure. Most spare tires are an example of this: they usually come with certain restrictions (e.g., a speed restriction) and lead to lower fuel economy. Another example is the "Safe Mode" found in most Windows operating systems.
- **Fail-safe systems** become safe when they cannot operate. Many medical systems fall into this category. For example, an infusion pump can fail, and, as long as it alerts the nurse and ceases pumping, it will not threaten the loss of life because its safety interval is long enough to permit a human response. In a similar vein, an industrial or domestic burner controller can fail, but must fail in a safe mode (i.e., turn combustion off when it detects a fault). Famously, nuclear weapon systems that launch-on-command are fail-safe, because if the communications systems fail, launch cannot be commanded. Railway signalling is also designed to be fail-safe.
- **Fail-secure systems** maintain security when they cannot operate. For example, while fail-safe electronic doors unlock during power failures, fail-secure ones will lock, keeping an area secure.
- **Fail-passive systems** continue to operate in the event of a system failure. An example includes an aircraft autopilot. In the event of a failure, the aircraft would remain in a controllable state and allow the pilot to take over, complete the journey, and perform a safe landing.

When safety-critical systems are connected to the cloud and safety-critical aspects either are distributed, or potentially affected by edge and cloud functionality, then provisions shall be taken to safeguard the safety of their operations. This section describes design concepts, real-time computing concepts, and dependable communication concepts for safety-critical platforms.

7.6.1 Design concepts for safety-critical operation

Overview

Design concepts for safety-critical operation are principles ensuring a correct system operation in spite of component failures (HW or SW), as well as communication failures between such components. Typically, safety analysis will assess the risk and impact of such failures and identify needed mitigations, requiring the use of such design concepts to increase safety integrity levels (SIL).

In (Armoush, 2010) design concepts for safety-critical embedded systems are catalogued and analysed. One such well-known concept is *triple modular redundancy (TMR),* which is typically applied in aircraft control systems. Triple modular redundancy provides protection against random faults and single point of failure. To that end, the TMR pattern provides an odd number of channels operating in parallel. The results (e.g., resulting actuation signals) are compared and, if the results differ in the three channels, a majority-vote is applied to determine the final result (Douglass, 2005). Triple modular redundancy is a symmetric concept, i.e., all parallel implementations are located in the device, although using heterogeneous HW in most cases.

Here, two safety concepts are highlighted which offer the potential for asymmetric application in device versus edge or cloud. These concepts require the system to have a *fail-safe state*: a reachable state that can

be considered safe in case of failure. In the cloud-edge-device continuum, on-device functionality shall be able to reach this fail-safe state in case the edge/cloud functionality fails.

The two highlighted safety concepts are the following:

- *Monitor-Actuator concept*: applicable for systems with a given fail-safe state and low availability requirements, where some level of redundancy is acceptable.
- *Safety Executive concept*: applicable for highly safety-critical systems in a situation requiring a complex shutdown of the actuation channel, with several safety-related actions to be coordinated.

These two concepts have promising applicability for a distributed realisation over the device-edge-cloud continuum of safety-relevant (mission-relevant) operation, redundancy, and monitoring. Further hardware and software design concepts, not highlighted in this document are: homogeneous or heterogeneous duplex channels; M-out-of-N channels, sanity-check, watchdog. In (Armoush, 2010) more information is provided on these concepts; in Figure 63 the watchdog concept is illustrated in the context of UC1.

Monitor-Actuator concept

The Monitor-Actuator concept (Armoush, 2010) is a special type of heterogeneous redundancy that is suitable for safety critical systems with low availability requirements and a fail-safe state, which is a condition of the system known to be always safe. This concept (see Figure 60) consists of two different channels (modules): A primary channel called the actuation channel, which performs the main action such as controlling some actuators, and a monitoring channel, which provides a monitoring for the actuation channel in order to detect and to identify the possible faults and then to make the actuation channel entering its fail-safe state.

The monitoring channel is used to improve the safety of the system by providing a continuous monitoring for the actuation channel to check its proper operation. The monitoring channel is independent from the actuation channel, so if there is a fault in the monitoring channel, then the actuation channel will continue to operate properly, but in this case any fault in the actuation channel cannot be detected.

The monitor takes the information about the outputs of the actuators, which is collected by the actuator sensors and processed by the monitoring acquisition system and compares it with the provided set points to make sure that the actuation channel is working properly. If the result of the comparison shows improper operation in the actuation channel, the monitor generates a shutdown signal to the actuation channel.

In the cloud-edge-device continuum, the Monitoring Channel are expected to reside on-device. The primary actuation channel can reside in the edge or cloud mostly, except for the actuation control. Shutdown of the actuation channel shall be sufficient to bring the system in a safe state.

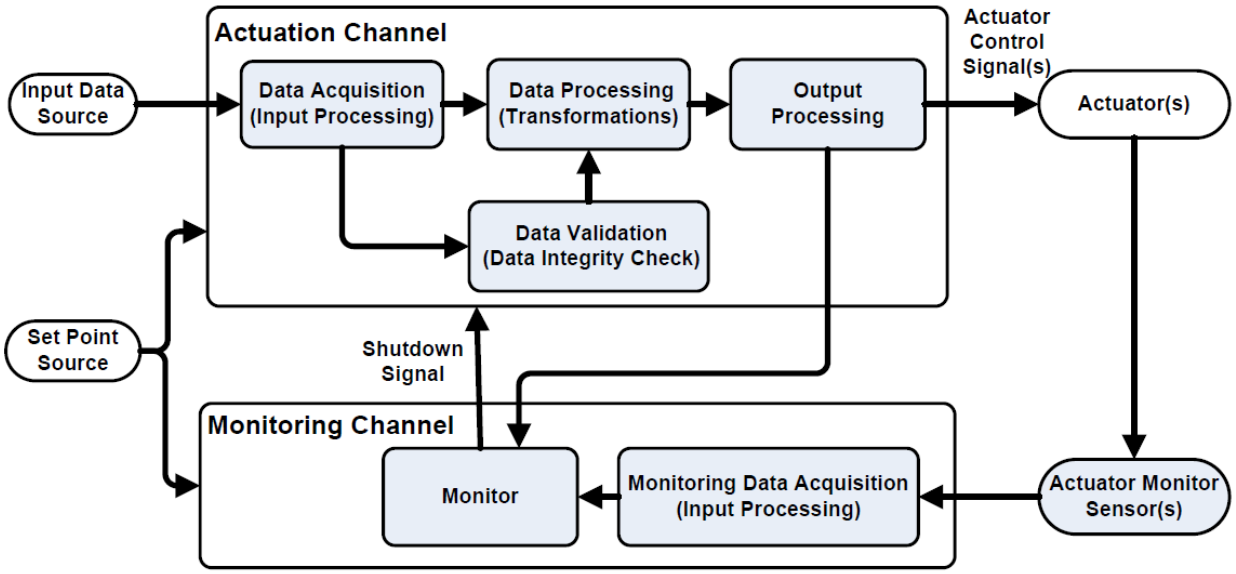

Figure 60: Monitor-Actuator concept (Armoush, 2010)

Safety Executive concept

The Safety Executive concept (Armoush, 2010) is a large-scale concept that is suitable for complex and highly safety-critical systems. It targets the problem where a shutdown of the system by the actuation channel itself might be critical or take too long time. This problem occurs especially in those systems in which a complicated series of steps involving several components is necessary to reach a fail-safe state. Therefore, the Safety Executive pattern uses a watchdog in combination with an additional safety executive component, which is responsible for the shutdown of the system as soon as the watchdog sends a shutdown signal (see Figure 61).

If the system has a fail-safe state, the safety executive component controls the shutdown of the actuation channels. Otherwise, the safety executive component has to conduct actions required to delegate an additional fail-safe processing channel to bring the system into a fail-safe state.

The Safety Executive concept provides a centralized and consistent method for monitoring and controlling the execution of a complex safety measure in case of failures.

The Safety Executive concept is based on an actuation channel to perform the required functionality and an optional fail-safe processing channel that is dedicated to the execution and control of the fail-safe processing. The central part of this concept is the existence of a centralized safety executive component coordinating all safety-measures required to shut down the system or to switch over to the fail-safe processing channel. The safety executive component can also be used to control multiple actuation channels in the system. The watchdog monitors the liveness of the actuation channel and, when compromised, signals a shutdown to the Safety Executive component.

The fail-safe processing channel is an optional channel dedicated to the execution and control of the fail-safe processing. In the presence of a fault in the actuation channel, the safety executive turns off the actuation channel, and the fail-safe processing channel takes over. If the system does not have a fail-safe channel, then the actuation channels must have at least one reachable fail-safe state.

In the cloud-edge-device continuum, the Safety Executive and its Fail-safe Processing Channel are expected to reside on-device. The primary Actuation Channel can reside in the edge or cloud mostly.

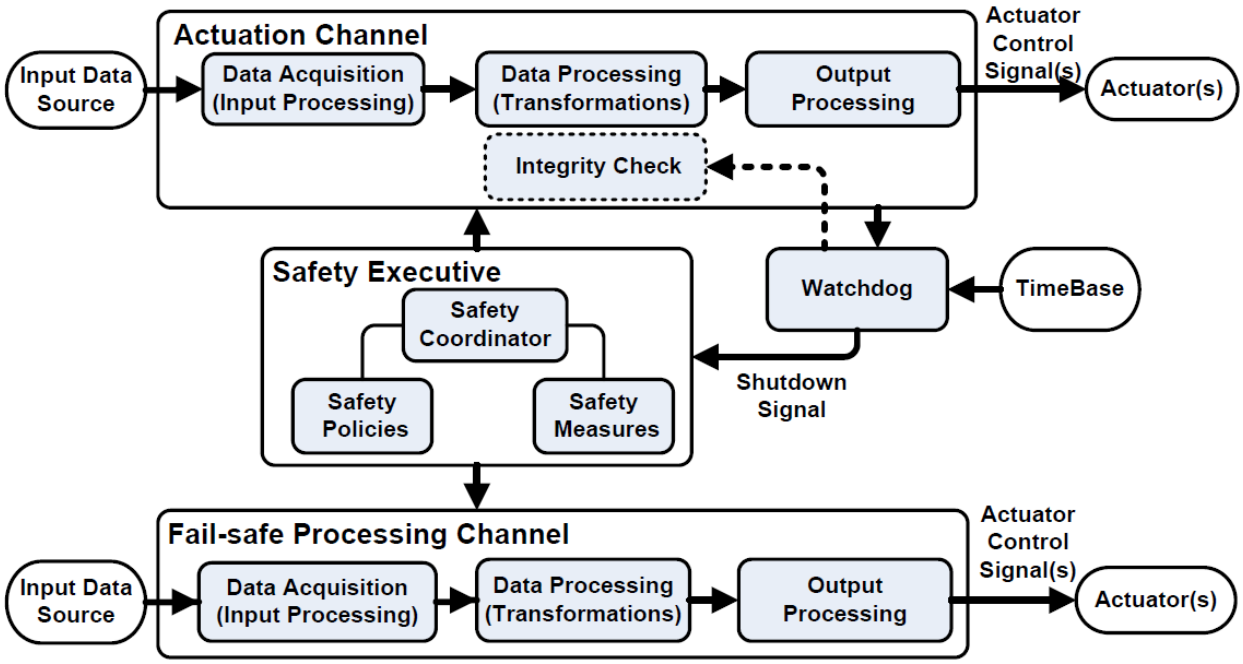

Figure 61: Safety Executive concept (Armoush, 2010)

Fit with concept TRANSACT reference architecture/components

In Figure 62, the application of these design concepts for *safety-critical* operations (blue circles) are positioned as concrete realisation of (parts of) the *Safety, Performance and Security Monitoring Service*, as well as in the *Operational Mode Manager* on the device tier and *Operational Mode Coordinator* on the edge tier. Although not shown explicitly, redundancy of the safety-critical function / mission-critical function may of course form part of the concept. For *mission-critical* applications (yellow circles), the actuation channel can also be extended to the cloud, for both monitoring services and operational mode coordination.

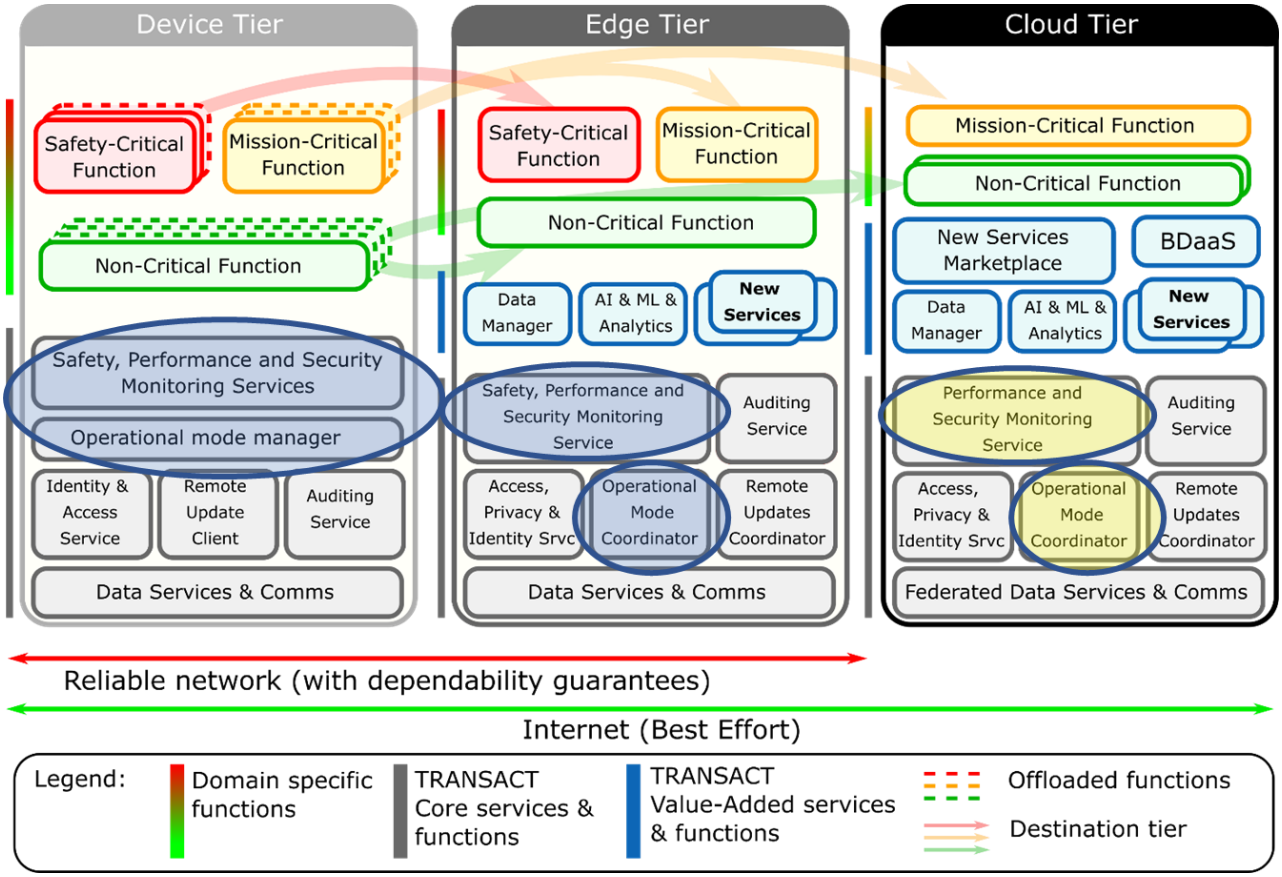

Figure 62: TRANSACT reference architecture: The design concepts are distributed over device and edge, for safety-critical use (blue), and can be extended to cloud in case of mission-critical use (yellow).

Example in context of a use case

In UC1, vehicle steering commands, such as "turn left 15 degrees", "brake pedal at position 0,2", "gas pedal at position 0,8", are sent to a remotely operated vehicle. In the vehicle software, a Watchdog pattern is implemented to ensure that the remote operated vehicle can be shut down gracefully in case the steering commands are not received anymore. An example sequence:

1. Vehicle receives a command "gas pedal at position 0.5"
2. Vehicle starts accelerating (actuators)
3. Watchdog time limit surpasses, i.e., no messages are received
4. Watchdog gives a shutdown signal to actuator channel
5. Vehicle stops to a halt (actuators)

Figure 63 below depicts the used watchdog pattern.

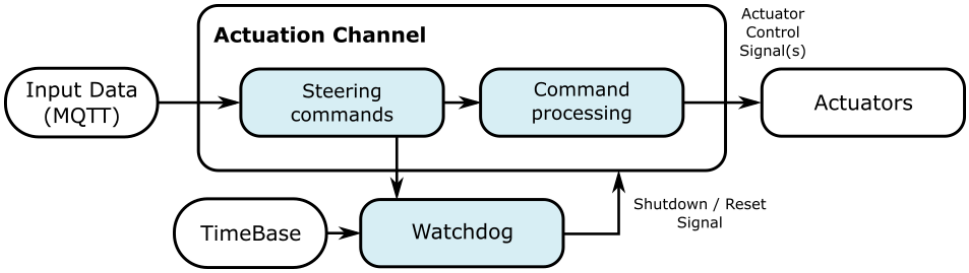

Figure 63: Watchdog pattern used in UC1 (Armoush, 2010).

Challenge for application within TRANSACT context

To date, safety-critical functionality and operations are almost exclusively positioned in the device. Mission-critical functionality is just started to be off-boarded. The key challenges in the TRANSACT context are the following:

- How to distribute the safety-relevant (mission-relevant) operation, redundancy, and monitoring over the device-edge-cloud continuum?
- What technologies (computing, communication, cloud) with which levels of dependability are needed to support such distribution of safety-critical functionality?
- How to keep safety (mission) concepts and safety proofs (safety cases) stable over edge or cloud-side upgrades to support faster innovation and independent releasing of new/upgraded edge or cloud functionality?

7.6.2 Real-time computing

Overview

Real-time systems are systems whose functions have timing requirements that must be satisfied to guarantee correct and safe operation. In many cases, this involves ensuring that a certain computation, such as computing the value of a process variable in a control system, is guaranteed to be completed within a given deadline, typically in the order of microseconds of milliseconds, with (very) high probability.

The execution time of a task on a processor is highly dependent on the underlying hardware, for a number of reasons (Dasari, Akesson, Nelis, Awan, & Petters, 2013). Many modern processors are optimized to provide high average performance in an energy-efficient manner through the use of multiple, possibly heterogeneous cores and complex memory hierarchies. However, the use of these processors typically implies big variations in task execution times (jitter), leading to poor worst-case performance (Heckmann, Langenbach, Thesing, & Wilhelm, 2003). To mitigate this, appropriate *hardware selection criteria* are essential to ensure that processors provide an appropriate balance between average and worst-case (guaranteed) performance and are amenable to either static or measurement-based timing analysis (Wilhelm, et al., 2008), if needed. As an example, very simple and time-predictable microcontrollers are often selected to execute safety-critical control tasks with strict timing requirements. In contrast, (heterogeneous) multi-core processors are selected for less critical tasks that may be more computationally intensive and require higher average performance, but where timing constraints may occasionally be violated.

System resources are often shared to manage qualities, such as cost, size, weight, and power. In a system with safety-critical functions, *spatial and/or temporal isolation* is required to mitigate interference between tasks sharing system resources and ensure safe operation. Spatial isolation implies mapping different

functions to different resources, such as processing cores or memory ranges, to reduce or eliminate interference. In contrast, temporal isolation is provided through scheduling that controls which tasks can use a shared resource at a particular time. Spatial and temporal isolation can be combined to create robust partitions, suitable for mixed-criticality systems with a mix of safety-critical, mission-critical, and non-critical functions (Ernst & Di Natale, 2016). Robust partitions based on spatial and temporal isolation are provided by a real-time operating system providing policies for mapping and scheduling. There are many different real-time operating systems and the choice for a particular system is often impacted by the domain. For example, real-time operating systems implementing the ARINC-653 specification (ARINC Incorporated, 2013) are often used in the avionics domain, where mitigation of interference is required for certification.

For mixed-criticality systems, *hardware virtualization* can additionally be provided by a hypervisor to enable multiple operating systems to be hosted on a single processor as virtual machines. This allows critical functions to be managed by a real-time operating system while mission-critical and non-critical functions can be managed by a standard operating system, such as Windows or Linux. This abstracts operating systems and applications from the underlying hardware, improving portability of applications and allowing resources to be efficiently shared.

Fit with concept TRANSACT reference architecture/components

The TRANSACT reference architecture considers mixed-criticality systems, as there is a mix of safety-critical, mission-critical, and non-critical functions, distributed over the three tiers, as shown in Figure 64.

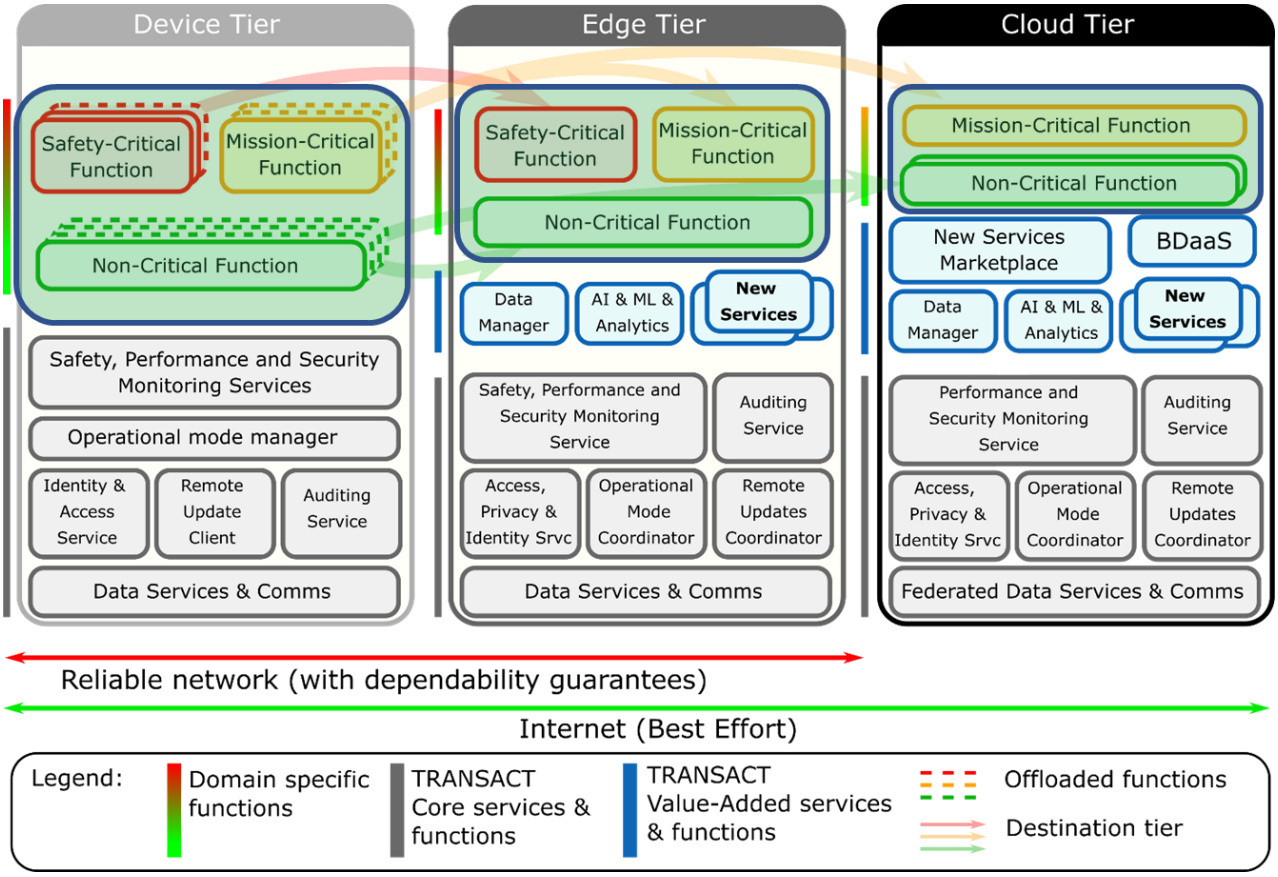

Figure 64: TRANSACT reference architecture: Real-time computing concepts require careful selection of hardware and software in each tier, depending on the criticality and timing requirements of their functions.

To enable real-time computing, hardware must be appropriately selected in each tier, depending on the criticality and timing requirements of the functions mapped to it. Hypervisors and real-time operating systems can be used to create robust partitions that spatially and temporally isolate functions (of same or different criticality) from each other, as necessary to fulfil safety requirements. In particular, it may be relevant to use isolation to separate mission-critical functions in the cloud from third-party applications and services downloaded from the marketplace.

Example in context of a use case

Timeliness of data and time constraints are important in UC1, when remotely operating autonomous vehicles. The remote driver receives a camera video stream from the vehicle camera. In order for the stream to be usable, the maximum camera-to-operator delay is targeted to be around 50-100ms. Delays larger than this limit the maximum usable speed of the vehicle considerably and reduce the safety of the remote driving operation.

To achieve this target performance, Fleetonomy.ai uses a Nvidia Jetson computer in order to encode and process the video stream on the GPU (onboard the vehicle). Video data is then transferred through a cellular network between the vehicle and the remote operation software - the cellular operator is selected based on the operation location to achieve optimal cellular reception and latency.

In other words, the aim is to minimize video delay in all parts of the device-edge-cloud continuum.

Challenge for application within TRANSACT context

- Real-time computing requires careful selection of hardware. While hardware in the device tier is generally appropriately selected for the needs of the particular system, this is less likely to be the case for the edge tier, which may support multiple (types of) systems. This limitation may be further exacerbated in cloud tier, which may be operated by a commercial provider and support systems from different organizations and domains. This limitation is likely to affect which functions that can be offloaded from the device. Functions with stringent timing requirements that must always be satisfied are unlikely to be offloaded, while functions with less strict requirements that may occasionally be violated can.
- The TRANSACT reference architecture is distributed, perhaps not only between tiers, but also within. This means that real-time computing must be combined with dependable communication, discussed next, to support safety-critical functions.

7.6.3 Dependable communication

Overview

In distributed Cyber-Physical Systems (CPS), communication between the different platforms is a key component. When safety-critical and mission-critical functions are off-loaded from constrained to more powerful devices, not only real-time computing concepts are required, but also the performance of data transmission between these platforms contributes to a successful (i.e., safe) execution of the task. Therefore, for safety-critical CPS, special emphasis should be placed on ensuring dependable (wired or wireless) communication, i.e., data exchange.

Dependable communication describes the reliability, timeliness, and availability of data transmission, i.e., it aims to minimize packet loss, latencies, and energy consumption. *Reliability* of communication protocols is typically measured in terms of packet reception rate (PRR). *Timeliness* often refers to the end-to-end latency between devices, and *availability* is typically linked to maximizing the system's operational time (e.g., by aiming for highly energy-efficient solutions), which is especially important for battery-powered devices (Schuß, Boano, Weber, & Römer , 2017).

The exact definitions of these dependability attributes may vary depending on the application's requirements. For example, reliability in distributed communication may also refer to fault-tolerance and continuity of correct service, i.e., ensuring that a (distributed) application can achieve its goal even in periods when parts of the communication system fail (Ivaki, Laranjeiro, & Araujo, 2018).

Compared to wired, wireless communication is exposed to different (environmental) factors that negatively affect end-to-end dependability. One issue is the radio hardware itself: for example, sensor devices are often shipped with low-gain antennas integrated on the board, which results in irregular radiation patterns complicating the operation of medium access control (MAC) and routing protocols. Environmental factors, such as interference, multi-path fading, and temperature, also impact the link quality and hence the communication performance of radio transceivers. Furthermore, wireless communication has asymmetric links, meaning that the link quality in one direction differs from the link in the other direction (Baccour, et al., 2012).

The following concepts are potentially interesting for the development of solutions to meet end-to-end dependability requirements in TRANSACT:

- *Monitoring and adaptation*: Link monitoring and measurements can be used to get an estimation of the link quality (e.g., observing PRR and latencies). Based on the link quality, the communication can be adapted. Adaptation of communication can, for example, be based on changing specific protocol parameters or using redundancy techniques to guarantee timely and reliable data transmission. Examples of such monitoring and adaptation concepts for wireless communication systems can be found, for instance, in (Baccour, et al., 2012), which provide a summary of related work on estimating link quality in wireless sensor networks (WSNs), (Boano, et al., 2009), which show the influence of transmit power adjustments, (Sha, Hackmann, & Lu, 2013) and (Boano, Römer, & Tsiftes, 2014), which adapt the Clear Channel Assessment (CCA) threshold, and (Park, Fischione, & Johansson, 2010) and (Zimmerling, Ferrari, Mottola, Voigt, & Thiele, 2012), which adjust other link-layer parameters to guarantee communication performance.

- *Time-triggered concepts*: They allow devices to exchange data in reserved time slots and therefore guarantee reliable communication. Such approaches, however, typically require precise time synchronization which is quite an overhead and may not be supported depending on the application.

- *Synchronous transmission*: A special concept for wireless communication are synchronous transmissions. They exploit the capture effect and constructive interference to effectively flood information in a network. Flooding-based solutions, such as Glossy, were shown to be reliable, while minimizing end-to-end latency and being energy-efficient (Zimmerling, Mottola, & Santini, Synchronous transmissions in low-power wireless: A survey of communication protocols and network services, 2020).

Fit with concept TRANSACT reference architecture/components

Dependable communication is required between each of the three tiers in the TRANSACT reference architecture. Each connection (i.e., device-edge, edge-cloud, and device-edge-cloud) will have more or less stringent end-to-end requirements depending on the use case: for example, when offloading safety-critical functions from the device to the edge, this connection has more stringent requirements than sending data for non-critical functions from the edge to the cloud. Because of the application-dependent requirements, several technologies may be used for data exchange, and therefore different concepts should be combined

to ensure dependable data transmission. In Figure 65, the blue arrows highlight that dependable communication affects the whole reference architecture.

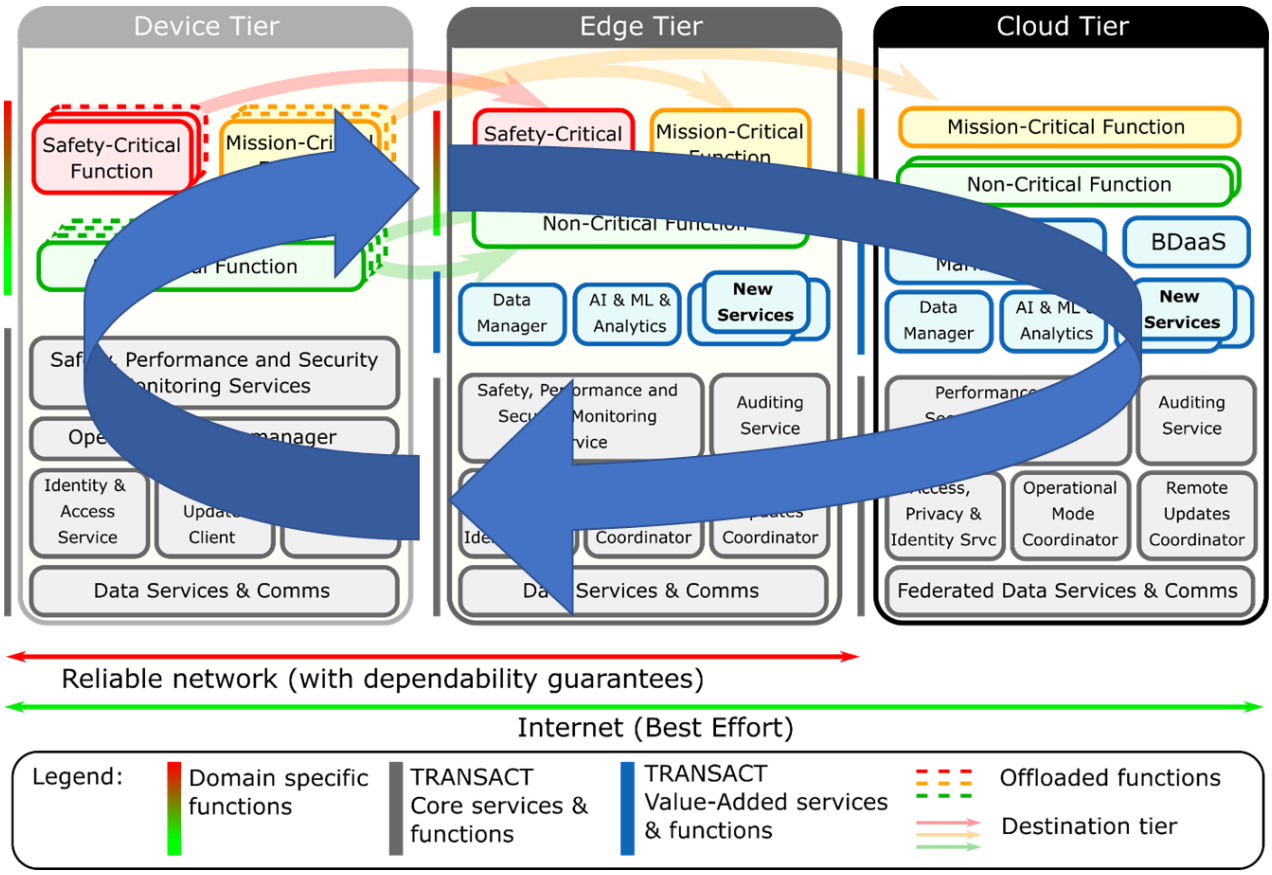

Figure 65: TRANSACT reference architecture: Dependable communication affects the whole device-edge-cloud continuum, i.e., each part of a platform that is involved in safety- and mission-critical data exchange.

Example in context of a use case

Every use case (UC) that depends on reliable and timely communication in the device-edge-cloud continuum to sustain the specified safety and performance goals, profits from these concepts.

In UC3, which addresses cloud-featured battery management systems (BMS), data from different vehicle components (e.g., battery and various sensors) is sent to a gateway (edge) and the cloud. This (sensitive) data is analysed and used for improvements on the vehicle's functionality (e.g., time left to charge), safety or autonomous driving (e.g., fail-operation in Battery/BMS). Several technologies will be involved in exchanging data between the different platforms: for example, CAN bus and Bluetooth Low Energy (BLE) for local communication inside the vehicle, and LTE for the link to the cloud. If BLE is used to transmit vehicle data from a component (sensor or actuator) to the gateway, interference with other devices operated in the same frequency band can cause message losses. Monitoring and adapting the BLE communication parameters can in this case help to achieve reliable and timely data transmission.

Dependable communication is a prerequisite for remotely operating autonomous vehicles in urban context (UC1). Communication between vehicle sensors and vehicle has to be reliable to form the best possible

understanding of the surrounding environment - this is needed when deciding whether to request transfer of control (ToC) to the remote driver. Video feed from the vehicle to the remote driver must be continuous (reliability) and with minimal latency (timeliness) in order for the remote driver to operate the vehicle safely. Environmental data (for example traffic control information) from external sources will be used to improve understanding of the world around the vehicle. This information also needs to be received with minimal latency in order for it the be usable.

Challenge for application within TRANSACT context

Concepts for dependable wireless and wired communication already exist, but the evaluation of their performance and applicability when different platforms are involved needs further investigation. Besides, there is always a trade-off between energy-efficiency and reliability. This creates additional challenges for platforms that need to sustain end-to-end reliability and end-to-end latency requirements, while preserving energy to maximize its lifetime.

The heterogeneity of technologies used for the data transmission between device, edge, and cloud makes maintaining the communication even more challenging. Existing concepts often provide solutions to achieve the best performance in a local network, but do not consider both device-edge and edge-cloud communication in their models.

8 Conclusion

8.1 Summary

This deliverable has presented selected concepts for end-to-end safety and performance for distributed CPS solutions. The selection of concepts has been done based on the TRANSACT use cases and their needs (see Section 3.1), as well as the technical requirements stemming from the TRANSACT WP1 analysis.

The selected concepts are clustered into three main categories: application concepts, cross-cutting concepts, and platform concepts as introduced in Section 4. For each of these categories, a number of concept classes have been identified. Within each concept class, then the selected concepts and methods are described.

The specific concepts in such a concept class can describe a further sub-division of the overall concept: e.g. in Section 6.4, Concepts for predictable performance encompasses the three sub-concepts Performance monitoring, Performance modelling and prediction, and Performance management respectively.

Alternatively, the specific concepts in a concept class may describe variations or alternatives of concepts to suit the needs of different domains and different characteristics: e.g. in Section 6.5, Concepts for safety and risk analysis, three different concepts for safety and risk analysis are included, each with a different scope and applicability in different lifecycle phases, to suit also the varying needs of the use cases and domains.

Finally, each concept is described using a homogenous structure. First the concept overview is presented, followed by how the particular concepts fits in the TRANSACT reference architecture. Then an example of application of the concept is given (in context of a particular use case), and lastly the challenges for application of the concept in the TRANSACT device-edge-cloud continuum type of systems are listed. These form the basis of further investigation in scope of the TRANSACT project.

8.2 Conclusions

One of the key objectives of the TRANSACT project is to *ensure safety and performance from an end-user perspective*. This requires firstly a thorough selection and evaluation of end-to-end safety and performance concepts for distributed safety-critical CPS solutions, and identify necessary existing or novel monitoring, analysis and run-time management techniques, including safety assurance concepts suitable for re-qualification. Once evaluated that these concepts meet this TRANSACT objective, they can be elaborated into solutions, which may be more specific to various domains and realisations of device-edge-cloud continuum systems and demonstrators.

The TRANSACT project task T3.1 has undertaken this selection and evaluation of concepts. The selected concepts are reported in this deliverable D3.1. The evaluation has reflected relevant safety and performance concerns, the TRANSACT reference architecture, and needs stemming from the use cases and WP1 analysis. These concepts now form the basis for further work to extend where necessary, and turn them into:

- Solutions to monitor and analyse end-to-end safety and performance, including the automatic generation of run-time monitors, model-based technique to analysis performance and resource contention, and requalification techniques.
- Solutions for (run-time) management for safety and performance, including online deployment optimization to minimize resource contention and activation mechanisms for operational safeguard and graceful degradation strategies.

Final validation of both these concepts and the corresponding TRANSACT elaborated solutions will be based on selected TRANSACT use case demonstrators created in TRANSACT WP5. These demonstrators will incorporate the selected concepts and solutions for validation.

9 References

(IEC), I. E. (2002). *Hazard and operability studies (HAZOP studies)—Application guide.*

Adelard. (2016). *ASCAD: The Adelard Safety Case Development Manual.* Adelard.

Ahlbrecht, A., & Durak, U. (2021). Integrating Safety into MBSE Processes with Formal Methods. *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)* (pp. 1-9). IEEE.

Albert Benveniste, B. C.-B.-V. (2012). *Contracts for System Design.* HAL Inria.

Albrecht, A., & Bertram, O. (2021). Evaluating System Architecture Safety in Early Phases of Development with MBSE and STPA. *2021 IEEE International Symposium on Systems Engineering (ISSE)* (pp. 1-8). IEEE.

AMASS. (2017). *AMASS Project Deliverables.* AMASS .

ARINC Incorporated. (2013). ARINC Specification 653.

Armoush, A. (2010). Design Patterns for safety-critical embedded systems. *Doctoral Dissertation*. Aachen, Germany: RWTH Aachen University.

Arnold, M., Boston, J., Desmond, M., Duesterwald, E., Elder, B., Murthi, A., . . . Reimer, D. (2020). Towards automating the AI operations lifecycle. *arXiv preprint, arXiv:2003.12808*. Retrieved from https://arxiv.org/pdf/2003.12808.pdf

AWS-monitoring. (2022, March 10). *Monitor Your Resources to Ensure That They Are Performing as Expected*. Retrieved from AWS: https://docs.aws.amazon.com/wellarchitected/latest/performance-efficiency-pillar/monitor-your-resources-to-ensure-that-they-are-performing-as-expected.html

AWS-scaling. (2022, March 10). *What is Amazon EC2 Auto Scaling?* Retrieved from AWS: https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html

AWS-scaling-and-balancing. (2022, March 10). *AWS-scaling-and-balancing, Amazon EC2 Auto Scaling benefits*. Retrieved from AWS: https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-benefits.html

Azim, A. C. (2014). Generation of communication schedules for multi-mode distributed real-time applications. *Design, Automation & Test in Europe Conference & Exhibition (DATE)* (pp. 1-6). IEEE.

Azure-availability-zones. (2022, March 10). *Cross-region replication in Azure: Business continuity and disaster recovery*. Retrieved from Azure documentation: https://docs.microsoft.com/en-us/azure/availability-zones/cross-region-replication-azure

Azure-monitoring. (2022, March 10). *Monitoring workloads*. Retrieved from Azure documentation: https://docs.microsoft.com/en-us/azure/architecture/framework/devops/monitor-pipeline

Baccour, N., Koubâa, A., Mottola, L., Zuniga, M. A., Youssef, H., Boano, C. A., & Alves, M. (2012). Radio Link Quality Estimation in Wireless Sensor Networks: a Survey. *ACM Transactions on Sensor Networks (TOSN)*, 1-33.

Baek, H. &. (2020). Response-Time Analysis for Multi-Mode Tasks in Real-Time Multiprocessor Systems. *IEEE Access, 8*, 86111-86129.

Balalaie, A., Heydarnoori, A., & Jamshidi, P. (2016). Microservices architecture enables devops: Migration to a cloud-native architecture. *IEEE Software 33(3)*, 42-52.

Bebawy, Y., Guissouma, H., Vander Maelen, S., Kröger, J., Hake, G., Stierand, I., . . . Hahn, A. (2020). Incremental Contract-based Verification of Software Updates for Safety-Critical Cyber-Physical Systems. *2020 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, United States, Dec 16, 2020 - Dec 18, 2020.*

Bellman, K., Landauer, C., Dutt, N., Esterle, L., Herkersdorft, A., Jantsch, A., & Tammemäe, K. (2020). Self-aware cyber-physical systems. *ACM Transactions on Cyber-Physical Systems 4(4)*, 1-26.

Benveniste A., C. B.-V. (November 2012). *Contracts for System Design.* Inria.

Benveniste, A. a.-B.-V. (2012). Contracts for Systems Design: Research Report. *hal-00757488*.

Berg, v. d., Čamra, V., Hendriks, M., Geilen, M., Hnetynka, P., Manteca, F., . . . Basten, T. (2020). QRML: A Component Language and Toolset for Quality and Resource Management. *Forum on specification & Design Languages, FDL 2020, Proceedings.* Kiel, Germany: IEEE Computer Society Press, Los Alamitos, CA, USA.

Boano, C. A., He, Z., Li, Y., Voigt, T., Zuniga, M., & Willig, A. (2009). Controllable radio interference for experimental and testing purposes in wireless sensor networks. *In Proceedings of the 2009 IEEE 34th Conference on Local Computer Networks* (pp. 865-872). IEEE.

Boano, C. A., Römer, K., & Tsiftes, N. (2014). Mitigating the Adverse Effects of Temperature on Low-Power Wireless Protocols. *In Proceedings of the 2014 IEEE 11th International Conference on Mobile Ad hoc and Sensor Systems (MASS)* (pp. 336-344). IEEE.

Burns, A. (2014). System mode changes-general and criticality-based. *2nd Workshop on Mixed Criticality Systems.*

Chakraborty, M., & Kundan, A. (2021). Architecture of a Modern Monitoring System. In *Monitoring Cloud-Native Applications* (pp. 55-96). Apress, Berkeley, CA.

Chen, T. &. (2018). SafeMC: A system for the design and evaluation of mode-change protocols. *In 2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 105-116.

Community, G. (2011). *GSN Community Standard Version 1.* Origin Consulting (York) Limited.

Dasari, D., Akesson, B., Nelis, V., Awan, M., & Petters, S. (2013). Identifying the sources of unpredictability in COTS-based multicore systems. *2013 8th IEEE international symposium on industrial embedded systems (SIES)* (pp. 39-48). IEEE.

Douglass, B. P. (2005). *Real-time design patterns: robust scalable architecture for real-time systems.* Addison-Wesley Professional.

Eclipse. (n.d.). *Eclipse PolarSys.* https://polarsys.org/ .

Ernst, R., & Di Natale, M. (2016). Mixed criticality systems - a history of misconceptions? *IEEE Design & Test, 3*, pp. 65--74.

Falcone, Y., Krstić, S., Reger, G., & Traytel, D. (2021). A taxonomy for classifying runtime verification tools. *International Journal on Software Tools for Technology Transfer, 23(2)*, 255-284.

Ferdous, R., Khan, F., Sadiq, R., Amyotte, P., & Veitch, B. (2013). nalyzing system safety and risks under uncertainty using a bow-tie diagram: An innovative approach. *Process Safety and Environmental Protection, 91*(1-2), 1-18. doi:10.1016/j.psep.2011.08.010

Ferretti, S., Ghini, V., Panzieri, F., Pellegrini, M., & Turrini, E. (2010). Qos–aware clouds. *IEEE 3rd International Conference on Cloud Computing* (pp. 321-328). IEEE.

Google. (2022, 03 11). *Anthos*. Retrieved from Google Cloud: https://cloud.google.com/anthos

Group, O. M. (2016). *SACM: Structured Assurance Case Metamodel. .* Object Management Group .

Guissouma, H., Kroger, J., Maelen, S. V., & Sax, E. (n.d.). Extension of Contracts for Variability Modeling and Incremental Update Checks of Cyber Physical Systems. *2021 IEEE International Symposium on Systems Engineering (ISSE). Hrsg.: Institute of Electrical and Electronics Engineers IEEE* (pp. 1–8). Institute of Electrical and Electronics Engineers (IEEE).

Heckmann, R., Langenbach, M., Thesing, S., & Wilhelm, R. (2003). The influence of processor architecture on the design and the results of WCET tools. *Proceedings of the IEEE* (pp. 1038-1054). IEEE.

Hendriks, M., Basten, T., Verriet, J., Brassé, M., & Somers, L. (2016). A Blueprint for System-Level Performance Modeling of Software-Intensive Embedded Systems. *Software Tools for Technology Transfer 18(1)*, 21-40.

Hendriks, M., Geilen, M., Goossens, K., de Jong, R., & Basten, T. (2021, May 26). Interface Modeling for Quality and Resource Management. *Logical Methods in Computer Science, LMCS*(19), 1-34. doi:10.23638/LMCS-17(2:19)2021

HIGHER COUNCIL FOR ELECTRONIC GOVERNMENT. (n.d.). *PORTAL ADMINISTRACIÓN ELECTRÓNICA.* Retrieved from https://administracionelectronica.gob.es/pae_Home/dam/jcr:80b16a91-75b1-432d-ab23-844a12aab5fc/MAGERIT_v_3_book_1_method_PDF_NIPO_630-14-162-0.pdf

IEC. (1998). *IEC 61508 Functional safety of electrical/electronic/programmable electronic safety-related systems.*

IEC. (2005). *Medical electrical equipment - Part 1: General requirements for basic safety and essential performance.* International Electrotechnical Commission (IEC).

IEC. (2010). *IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems.* IEC.

IEC. (2016). *Health software - Part 1: General requirements for product safety .* IEC.

IEC. (2016). *IEC 61511 Functional safety - Safety instrumented systems for the process industry sector.*

ISO. (2005). *ISO 17894:2005, Ships and marine technology.*

ISO. (2009). *ISO 26262 - Road vehicles—Functional safety. International Organization for Standardization / Technical Committee 22.* ISO TC 22.

ISO. (2011). *ISO 26262 - Road vehicles -- Functional safety.* ISO, Geneva, Switzerland.

ISO. (Under development). *ISO/FDIS 21448 - Road vehicles — Safety of the intended functionality.* ISO.

Ivaki, N., Laranjeiro, N., & Araujo, F. (2018). A survey on reliable distributed communication. *Journal of Systems and Software*, 713-732.

Kalra, N. P. (2016). Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part 94*, pp. 182–193.

Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. *Computer, 36(1)*, 41-50.

Kienhuis, B., Deprettere, E., Vissers, K., & Wolf, v. d. (1997). An Approach for Quantitative Analysis of Application-Specific Dataflow Architectures. *Proc. IEEE Int. Conf. on Application-Specific Systems, Architectures and Processors, ASAP'97* (pp. 338–349). IEEE.

Kornaros, G., & Pnevmatikatos, D. (2013). A survey and taxonomy of on-chip monitoring of multicore systems-on-chip. *ACM Transactions on Design Automation of Electronic Systems (TODAES), 18(2)*, 1-38.

Kramer, B., Neurohr, C., Büker, M., Böde, E., Fränzle, M., & Damm, W. (2020). Identification and Quantification of Hazardous Scenarios for Automated Driving. *Model-Based Safety and Assessment, 7th International Symposium, IMBSA 2020*, (pp. 163-178).

Lapalme, J., Theelen, B., Stoimenov, N., Voeten, J., Thiele, L., & Aboulhamid, E. (2009). Y-chart Based System Design: a Discussion on Approaches. In *Nouvelles approches pour la conception d'outils CAO pour le domaine des systems embarqués.* Université de Montreal.

Lee, K., Lee, K., Lee, H., & Shin, J. (2018). A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems, 31*.

Leveson, N. G. (2016). *Engineering a safer world: Systems thinking applied to safety.* Boston: The MIT Press.

MAGERIT. (2005). Retrieved from European Union Agency for Cybersecurity: https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory/rm-ra-methods/m_magerit.html

Martin, H. (November 2018). *AMASS Deliverable D6.8 "Methodological guide for cross/intra-domain reuse (b).* ECSEL Research and Innovation actions (RIA).

Meyer, B. (1986). Design by Contract. Technical Report TR-EI-12/CO. *Interactive Software Engineering Inc*.

Park, P., Fischione, C., & Johansson, K. H. (2010). Adaptive IEEE 802.15.4 Protocol for Energy Efficient, Reliable and Timely Communications. *In Proceedings of the 9th ACM/IEEE international conference on information processing in sensor networks* (pp. 327-338). ACM.

PEGASUS Project. (2022, 26 04). Retrieved from https://www.pegasusprojekt.de/de/

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should i trust you?" Explaining the predictions of any classifier. *22nd ACM SIGKDD international conference on knowledge discovery and data mining*, (pp. 1135-1144). Retrieved from https://arxiv.org/pdf/1602.04938.pdf

RTCA. (1992). *RTCA DO-178B. Software Considerations in Airborne Systems and Equipment Certification.* Radio Technical Commission for Aeronautics (RTCA).

Rushby, J. (2008). Runtime Certification. *International Workshop on Runtime Verification.* Budapest, Hungary: Springer.

Sáez, S. R. (2012). An integrated framework for multiprocessor, multimoded real-time applications. *International Conference on Reliable Software Technologies* (pp. 18-34). Springer.

Safety Engineering. (2022, February 14). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Safety_engineering

Samek, W., Wiegand, T., & Müller, K. R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint, arXiv:1708.08296*. Retrieved from https://arxiv.org/pdf/1708.08296.pdf

Sanden, v. d., Hi, Y., Aker, v. d., Akesson, B., Bijlsma, T., Hendriks, M., . . . Basten, T. (2021). Model-driven system-performance engineering for cyber-physical systems. *International Conference on Embedded Software (EMSOFT)* (pp. 11-22). ACM.

Sau, C., Rinaldi, C., Pomante, L., Palumbo, F., Valente, G., & Fanni, T. (2021, November). Design and Management of Image Processing Pipelines within CPS: Acquired Experience towards the end of

the FitOptiVis ECSEL Project. *Microprocessors and Microsystems: Embedded Hardware Design, MICPRO*(87).

Schuß, M., Boano, C. A., Weber, M., & Römer , K. (2017). A Competition to Push the Dependability of Low-Power Wireless Protocols to the Edge. *International Conference on Embedded Wireless Systems and Networks (EWSN '17)* (pp. 54-65). Uppsala, Sweden: Junction Publishing.

Schuß, M., Boano, C. A., Weber, M., & Römer , K. (2017). A Competition to Push the Dependability of Low-Power Wireless Protocols to the Edge. *In Proceedings of the 14th International Conference on Embedded Wireless Systems and Networks (EWSN '17)* (pp. 54-65). Junction Publishing.

Sha, M., Hackmann, G., & Lu, C. (2013). Energy-efficient low power listening for wireless sensor networks in noisy environments. *In Proceedings of the 12th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)* (pp. 277-288). IEEE.

Stantchev, V., & Schröpfer, C. (2009). Negotiating and Enforcing QoS and SLAs in Grid and Cloud Computing. *International Conference on Grid and Pervasive Computing*, 25-35.

Step-Up!CPS. (2021, 12 15). *Step-UP!CPS*. Retrieved from https://stepup-cps.de/

Strathmann, T., Hake, G., Guissouma, H., Hohl, C. P., Bebawy, Y., Maelen, S. V., & Koerner, A. (2021). Project Overview for Step-Up!CPS - Process, Methods and Technologies for Updating Safety-critical Cyber-physical Systems. *Design, Automation & Test in Europe (DATE 2021 2021)* (pp. 1326-1329). Institute of Electrical and Electronics Engineers (IEEE).

Two-dimensional filter. (2021, March 13). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Two-dimensional_filter

UNION, T. E. (5 April 2017). *REGULATION (EU) 2017/745 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL.* THE EUROPEAN PARLIAMENT AND THE COUNCIL OF THE EUROPEAN UNION.

Valente, G., Fanni, T., Sau, C., Mascio, T., Pomante, L., & Palumbo, F. (2021). A composable monitoring system for heterogeneous embedded platforms. *ACM Transactions on Embedded Computing Systems (TECS), 20(5)*, 1-34.

van Amersfoort, J., Smith, L., Jesson, A., Key, O., & Gal, Y. (2021). On feature collapse and deep kernel learning for single forward pass uncertainty. *arXiv preprint, arXiv:2102.11409*. Retrieved from https://arxiv.org/abs/2102.11409

Wikipedia. (2021, November 14). *Safety-critical system*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Safety-critical_system

Wilhelm, R., Engblom, J., Ermedahl, A., Holsti, N., Thesing, S., Whalley, D., . . . Stenstrom, P. (2008). The worst-case execution-time problem—overview of methods and survey of tools. *ACM Transactions on Embedded Computing Systems (TECS), 7*(3), pp. 1-53.

Yu, M., Wu, C., & Tsung, F. (2019). Monitoring the data quality of data streams using a two-step control scheme. *IISE Transactions, 51*(9), 985-999.

Zeller, M. (2021). Towards Continuous Safety Assessment in Context of DevOps. *arXiv preprint arXiv:2106.07200*. Retrieved from arXiv preprint: https://arxiv.org/pdf/2106.07200.pdf

Zhang, W., & Mei, Y. (2020). Bandit Change-Point Detection for Real-Time Monitoring High-Dimensional Data Under Sampling Control. *arXiv preprint, arXiv:2009.11891*.

Zimmerling, M., Ferrari, F., Mottola, L., Voigt, T., & Thiele, L. (2012). pTunes: Runtime Parameter Adaptation for low-power MAC Protocols. *In Proceedings of the 11th international conference on Information Processing in Sensor Networks* (pp. 173-184). IEEE.

Zimmerling, M., Mottola, L., & Santini, S. (2020). Synchronous transmissions in low-power wireless: A survey of communication protocols and network services. *ACM Computing Surveys (CSUR)*, 1-39.