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

D16 (D3.3)

Solutions for end-to-end safety and performance for distributed CPS v1

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.

Document Information

Project	TRANSACT
Grant Agreement No.	101007260
Work Package No.	WP3
Task No.	T3.3
Deliverable No.	D16
Deliverable No. in WP	D3.3
Deliverable Title	Solutions for end-to-end safety and performance for distributed CPS
Nature	Report
Dissemination Level	Public
Document Version	v1.0
Date	30/11/2022
Contact	Mitra Nasri
Organization	TUE (Eindhoven University of Technology)
Phone	+31 626466032
E-Mail	m.nasri@tue.nl

Authors Table

Name	Company	E-Mail
Mitra Nasri	TUE	m.nasri@tue.nl
Martijn Hendriks	TUE	m.hendriks@tue.nl
Marc Geilen	TUE	m.c.w.geilen@tue.nl
Jeroen Voeten	TUE	j.p.m.voeten@tue.nl
Twan Basten	TUE	a.a.basten@tue.nl
Nasim Samimi Dehkordi	TUE	n.samimi.dehkordi@tue.nl
Seyed Ali Kiaian Mousavy	TUE	s.a.kiaian.mousavy@tue.nl
Benny Akesson	TNO	benny.akesson@tno.nl
Teun Hendriks	TNO	teun.hendriks@tno.nl
Miguel García Gordillo	ITI	miguelgarcia@iti.es
Joan J. Valls Mompó	ITI	jvalls@iti.es
Kilian Michiels	FEops	kilian.michiels@feops.com
Thomas Netsch	PFLH	thomas.netsch@philips.com
Dusica Marijan	SRL	dusica@simula.no
Tetyana Kholodna	NVT	Tetyana.kholodna@navtor.com
Elisabeth Salomon	TUG	elisabeth.salomon@tugraz.at
Sara Pastor	SNG	Sara.pastor@nunsys.com
Gaurav Choudhary	DTU	gauch@dtu.dk
Paul Pop	DTU	paupo@dtu.dk
Nicola Dragoni	DTU	ndra@dtu.dk
Sara Pastor	SNG	sara.pastor@nunsys.com
Damian Aparicio	SNG	damian.aparicio@nunsys.com
Koen de Laat	PMS	koen.de.laat@philips.com

Reviewers Table

Version	Date	Reviewer
0.4	29.09.2022	Martijn Hendriks (TUE)
0.4	29.09.2022	Gaurav Choudhary (DTU)
0.6	19.10.2022	Sebastian Vander Maelen (DLR)
0.6	03.11.2022	Ken Sharman (ITI)
0.6	07.11.2022	Marco Jahn (ECL)
0.7	29.11.2022	Sasa Marinkovic (PMS)

Change History

Version	Date	Reason for Change	Affected pages
0.4	13.10.2022	Applying feedback of the internal reviewers	All
0.6	10.11.2022	Annotated with feedback of external reviewers	All
0.7	15.11.2022	Applied feedback of external reviewers	All
1.0	30.11.2022	Final version for submission	n/a

Table of Contents

List of Figures

List of Tables

List of participants

Part no.	Participant organisation name	Participant short name	Country
1	Philips Medical Systems Nederland BV	PMS	NL
2	Technische Universiteit Eindhoven	TUE	NL
3	PS-Tech BV	PST	NL
4	Vinotion BV	VIN	NL
5	Nederlandse Organisatie voor Toegepast Natuurwetenschappelijk Onderzoek	TNO	NL
6	Fleetonomy.ai Oy	FLEET	FI
7	Teknologian tutkimuskeskus VTT Oy	VTT	FI
8	F-Secure Oyj	FSC	FI
9	Nodeon Finland Oy	NOD	FI
10	AVL Software and Functions GmbH	AVL	DE
11	Eclipse Foundation Europe GmbH	ECL	DE
12	OFFIS E.V.	OFFIS	DE
13	Philips GmbH	PFLH	DE
14	Denso Automotive Deutschland GmbH	DNDE	DE
15	Fraunhofer-Gesellschaft - Fraunhofer Institute for Experimental Software Engineering	IESE	DE
16	Politechnika Gdanska	GUT	PL
17	DAC Spolka Akcyjna	DAC	PL
18	Technische Universitaet Graz	TUG	AT
19	CISC Semiconductor GmbH	CISC	AT
20	NAVTOR AS	NVT	NO
21	SIMULA Research Laboratory AS	SRL	NO
22	Instituto Tecnologico de Informatica	ITI	ES
23	NUNSYS SL	NUN	ES
24	Kumori Systems	KUM	ES
25	SINGLAR INNOVACION SL	SNG	ES
26	Fundació per a la Universitat Oberta de Catalunya	UOC	ES
27	Depuración de Aguas del Mediterráneo SL	DAM	ES

28	FEops NV	FEops	BE
29	Danmarks Tekniske Universitet	DTU	DK
30	Toitware ApS	TW	DK
31	Deutsches Zentrum fur Luft- und Raumfahrt EV	DLR	DE

Table 1: List of TRANSACT participants

Glossary

Term	Definition
AWS	Amazon Web Services
BLE	Bluetooth Low Energy
CP	Constraint Programming
DR	Disaster Recovery
ECP	Edge Computing Platforms
EDs	Edge Devices
FSM	Finite State Machine
GCL	Gate-Control List
IoT	Internet-of-Things
IT	Information Technology
ITSCM	IT Service Continuity Management
ITIL	IT Infrastructure Library
MAC	Message Authentication Code
MCR	Mode Change Request
MCP	Mode Change Protocol
OGC	Office of Government Commerce
PoC	Proof of Concept
RA	Remote Attestation
SLO	Service Level Objective
SP	Strict Priority
STAMP	Systems-Theoretic Accident Model and Processes
STPA	Systems-Theoretic Process Analysis
TSN	Time-Sensitive Networking
XSTAMPP	eXtensible STAMP Platform

Table 2: Terms, Abbreviations and Definitions

1 Introduction

This report is a result of Task T3.3 of the TRANSACT project. This task was to develop solutions to monitor, predict and manage end-to-end safety and performance of distributed applications, including solutions for (re)qualification. In particular, Task T3.3 contributes mainly to the Objective 3 of the TRANSACT project (which focuses on ensuring safety and timing performance from a user perspective) by developing:

- **Obj A**: Solutions to monitor and analyze end-to-end safety and performance, including the automatic generation of run-time monitors, model-based techniques to analysis performance and resource contention, and safety assurance concepts/frameworks suitable for re-qualification.
- **Obj B**: 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.

In addition, Task T3.3 also partly contributes to Objective 1 of the TRANSACT project (which focuses on leveraging edge and cloud technologies for safety-critical CPSs) by delivering:

- **Obj C**: Self-monitoring techniques for learning systems
- **Obj D**: Solutions and architectures for scalable applications and platforms, as well as runtime scaling strategies

In the rest of this section, we first describe the purpose of the deliverable and its relation with other deliverables of the TRANSACT project (Section 1.1). Then we will focus on how the solutions described in this deliverable contribute to the four objectives, i.e., **Obj A, B, C,** and **D**, and therefore to the Objectives 1 and 3 of the TRANSACT project (Section 1.2.1). Moreover, we provide a discussion on the relation/interaction between techniques developed to improve/guarantee performance and safety and those for security and privacy (delivered by Task T3.4) in Section 6. Finally, we describe how the deliverable is structured in a way that it is consistent with the outputs of Task 3.1 (concepts for safety and performance in safety-critical CPSs).

1.1 Purpose of the deliverable and its relation to other deliverables

The following are the major purposes of this document:

- Documentation of selected solutions for safety and performance analysis, to be applicable across the TRANSACT domains.
- Documentation of selected solutions 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.

We have aligned the contents and results of this deliverable with the 'sister' deliverable D3.4 (from Task T3.4) which targets solutions for security and privacy for distributed CPS. We ensured that the outcome of this task fits with the TRANSACT reference architectures as described in Deliverable D2.1. More precisely, each of the solutions introduced in the document have a clear connection to the TRANSACT reference architecture and a description that tells how they are related to or can be used by various use cases of the TRANSACT project.

This document relates to the following TRANSACT deliverables:

- D5 (D1.1) Use case descriptions, end user requirements, state of the art and KPI's (M10)

The selected solutions 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 discuss how they can help or be applied on different use cases of the project.

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

  The developed solutions 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.

- D9 (D3.1) Selection of concepts for end-to-end safety and performance for distributed CPS solutions (M12)

  The solutions developed in this deliverable are realization of the concepts for end-to-end safety and performance for distributed safety-critical CPS that have been introduced in Deliverable D3.1. We follow the same 3-layer categorization (and hence have application-related, platform-related, and cross-cutting solutions that realize the same concepts from D3.1).

- D9 (D3.2, M12) Selection of concepts and D17 (D3.4, M18) Selection of solutions for end-to-end security and privacy for distributed CPS solutions

  The solutions developed for end-to-end safety and performance for distributed safety-critical CPS are harmonised with the complementary concepts and solutions for end-to-end security and privacy for distributed CPS solutions as documented in D3.2 and D3.4. We have provided a section in this document (Section 6) to explain the relation between these concepts/solutions by looking at the impact of security risks/threats on safety and performance (and vice versa) and the alignment between the solutions for security and privacy and safety and performance (both positive and negative impact of these solutions on one another).

1.2 Role of the deliverable

Task T3.3 has two deliverables: D3.3 (due in M18) and D3.5 (due in M33). The purpose of this breakdown is to allow solutions that require further investigation or are tightly coupled with the use cases enough time to mature. Some of those solutions will appear only in D3.5 while some other will have two versions: the primary version will be presented in D3.3 and an extended version will be presented in D3.5.

Table 3 lists the solution items developed in D3.3 and shows how they contribute to the four objectives, i.e., **Obj A, B, C,** and **D**, and therefore to the Objectives 1 and 3 of the TRANSACT project.

Solution	Title	Section	Objective 3 (main focus)		Objective 1 (secondary focus)	
			Obj A	Obj B	Obj C	Obj D
S1	Mode change management on the device	Section 3.2.1		●		
S2	Mode change coordination	Section 3.2.2		●		
S3	Solutions for scalable applications	Section 3.3				●
S4	Solutions for AI-monitoring	Section 3.4			●	
S5	Solutions for ensuring data integrity	Section 3.5			●	●
S6	Performance observability and monitoring	Section 4.2.1.1	●			
S7	AI-based performance modeling and prediction	Section 4.2.2.1	●			

S8	Simulation-based performance analysis	Section 4.2.2.2	●			
S9	Performance analysis using formal methods	Section 4.2.2.3	●			
S10	Workflow simulation	Section 4.2.2.4	●			
S11	Scenario-based performance management and reconfiguration	Section 4.2.3.1		●		
S12	Risk management planning/monitoring	Section 4.3.1	●			
S13	Real-time machine-learning based solutions for detecting safety, security, and privacy anomalies	Section 4.3.2	●	●		
S14	Mapping and scheduling techniques across device, edge, and cloud	Section 4.3.3		●		
S15	Service continuity monitoring	Section 0	●			
S16	Solutions for dependable wireless communication	Section 5.2.1				●
S17	Solutions for scalable platforms, and run-time scaling strategies	Section 5.3		●		●

Table 3: List of solution items presented by the Deliverable D3.3

1.2.1 Mapping between the objectives and developed solutions

In particular, to achieve **Obj A**, deliverable D3.3 includes

- Development of monitoring tools:
  - S6 investigates available monitoring tools/techniques for edge/cloud platforms that are able to gather a wide range of performance- and safety-related metrics;
  - S12 and S13 provide another monitoring tool specialized to gather safety- and security-related data from the system;
  - S15 discusses monitoring techniques/tools for observing service availability across the device-edge-cloud continuum;
- Development of model-based or data-driven performance prediction/analysis techniques:
  - S7 and S8 investigate AI-based and simulation-based performance prediction methods;
  - S9 provides a formal analysis to obtain the worst-case performance applications running on the device-edge-cloud continuum;
  - S10 provides a simulation-based analysis to study and predict the performance of workflows in healthcare applications;
  - S13: applies machine-learning-based techniques to detect safety and security anomalies
- Development/application of safety assurance concepts/frameworks:
  - S12 introduces a unified risk-analysis framework for both safety and security risks. The framework is based on MARGERIT methodology, which is a standard that establishes principles for the effective, efficient, and acceptable use of IT, prepared by the CSAE (Spanish Higher Council of E-Government) and published by the Ministry of Finance and Public

Administrations of Spain. It includes safety requirements of various application domains (via the standards applied on those domains). Using the guidelines of those standards, it provides an extensive set of risks (related to safety and security of the system) that must be taken into account before a safety case can be made. Risk assessment analysis methods will be part of an overall TRANSACT safety concept that will be presented in detail the year 2 periodic report.

To achieve **Obj B**, Deliverable D3.3 includes

- Solutions for online deployment optimization:
  - S11 presents scenario-based performance management and reconfiguration solutions that can be used for finding optimal configurations/setup for deployment;
  - S15 suggests monitoring approaches to check adequacy of deployment and mapping, choices across device, edge, and cloud continuum;
- Solutions for operational safeguard and strategies for graceful degradation:
  - S1 and S2 present techniques for application mode change (including the transition to a degraded mode of operation) and mode-change coordination;
  - S11 and S17 provide techniques for runtime resource management. In Deliverable D3.5, S11 will be extended to include a wider variety of resource-management techniques, mode changes, and fallback scenarios;
  - S13 provides mechanism to detect safety and security anomalies and raise alarm (which could be used to trigger fall-back scenarios);

To achieve **Obj C**, Deliverable D3.3 includes

- Solutions for monitoring performance of AI-based applications to ensure their quality and keep them tuned and updated as new data enters the system (S4);
- Solutions for ensuring the integrity of data (S5).

To achieve **Obj D**, Deliverable D3.3 includes

- Solutions for scalable applications (S3);
- Solutions for scalable platforms (cloud/edge) via S17 (which explores a collection of existing techniques to transform device-based CPSs to cloud-based CPSs);
- Solutions for scalable, high-performance, and dependable networks via S14 (which investigates and develops new techniques for increasing the dependability of wireless communication for safety-critical CPSs).

(Re)qualification and (re)certification are tightly related to safety assurance and safety cases (made for the safety standards that are applicable to the application domain, e.g., automotive, healthcare, etc.). Before such cases can be made, there is a need for a thorough safety and risk analysis of the system (before and after) it is moved to the cloud/edge continuum. The role of Deliverable D3.3 is to provide solutions/frameworks for risk-analysis and safety assurance. As explained earlier, these solutions will be part of an overall TRANSACT safety assurance concept, which will be presented in the Y2 periodic report. One of the ingredients of the overall safety assurance concepts is risk assessment, which will be addressed via S12 (which applies MARGERIT methodology). The decision to use this tool was the conclusion of a set of project-wide activities:

- DTU (in collaboration with external experts, TECNALIA, who coordinated AMASS project) provided a tutorial with the main results of AMASS, with a focus on modelling the assurance process needed for specific functional safety standards using the open source OpenCert tool (during the Assembly Meeting of the TRANSACT project in June 2022). This workshop was especially designed to be accessible for the use-case owners, letting them get familiar with the PolarSys environment and AMASS framework. In addition to knowledge transfer, this workshop raised a lot of interesting discussions and interactions among partners during and after the assembly meeting.
- The TRANSACT use cases have presented their safety and security assurance challenges. The outcome was that the challenges the use cases face are at the intersection of security and safety. That is, how would security threats, which are increasing in likelihood with edge-cloud-based solutions, jeopardize the safety. The conclusion was that the use cases would benefit more from performing a joint risk analysis of security and safety risks and their interplay. This conclusion is also aligned with the preliminary recommendation from the first review meeting, which recommended that the use cases look into security risks related to transitioning from monolithic systems to edge-based solutions in TRANSACT.
- In July 2022, Task T3.3 held another workshop in which SNG presented GConsulting risk-analysis and safety-assurance tool which is based on the MARGERIT methodology. To have uniformity in the tools used for risk analysis and safety assurance, a number of partners decided to apply the MARGERIT methodology on the use cases as it already included a set of relevant standards and their safety/security requirements.

The project is now evaluating approaches to hazard and risk assessment, e.g., the MAGERIT approach (with corresponding tools) provided by the project partners Singlar and Nunsys, or the STAMP (Systems-Theoretic Accident Model and Processes)/ STPA (Systems-Theoretic Process Analysis) hazard and risk analysis method supported by tools such as XSTAMPP (An eXtensible STAMP Platform As Tool Support for Safety Engineering). These risk assessment approaches will be part of the overall TRANSACT safety concept, which will be presented in the Y2 deliverables.

Furthermore, depending on the outputs of the risk analysis stage (which will determine the safety-integrity level of each use case), different (re)qualification actions/solutions might be needed. These actions/solutions will be completed/provided in both Deliverables D3.3 and D3.5 and the alignment meetings and workshops throughout the project.

1.2.2 Mapping between the developed solutions and the technical system requirements (TSR)

In this section, we briefly discuss how the solutions developed in this deliverable could potentially contribute to the technical system requirements for the TRANSACT system level (TSR) gathered and summarized in Deliverable D1.2.

Deliverable D1.2 first gathers the requirements of use cases and then summarizes them into a set of technical system requirements presented in section 6.9.2 of D1.2. Some of these requirements are satisfied by the TRANSACT architecture or solutions/concepts developed in other tasks and work packages. To keep this section brief, we only focus on TSRs that could be addressed by the proposed solutions in Task T3.3 (in Table 4).

TSR No	Description	Potentially relevant safety/performance solutions identified in Table 3
TSR 21	In light of the diversity of application architectures and cloud infrastructures, many types of metrics have been considered as optimization objectives during the behavior modelling and resource provisioning of containerized applications, including (1) resource efficiency, (2) energy efficiency, (3) cost efficiency, and (4) SLA assurance.	S11 (and its extension in D3.5)
TSR 23	The cloud systems when used by the architecture in the Use Cases (see TSR 59) should meet the following requirements: (1) No data loss: The system must ensure with a high probability that data will not be lost permanently.	S5
	(2) High availability: Data must be available to users/Use Cases when they want the data with a reasonably high probability, though some occasional temporary outages are acceptable.	S6-S9, S13, S15
	(3) High performance: The system should not perform significantly worse than the current usual alternatives notably NFS (Network File System).	S3, S17
	(4) Scalability: The system must scale to large numbers of clients, large numbers of storage 'nodes', large aggregate storage spaces, etc.	S3, S16, S17
	(5) Cost efficiency: Since there are existing solutions to buy large, reliable storage, the system must be inexpensive in hardware, software and maintenance.	S8 (and its extension in D3.5) S17
	(6) Security: The system must be able to match the confidentiality, data integrity and authorization standards expected by users/Use Cases.	S5, S13
TSR 24	The cloud systems (when used in the Use Cases) and their provider used by the architecture in cooperation with the Use Case should comply with existing regulatory requirements and Service Level SLA.	S12, S13
TSR 32	For real-time applications and near real time applications, immediate data analysis shall be required for the architecture. Therefore, new technologies, such as edge computing, should be fused with IoT/sensor networks.	S6 and S13 used together with S7, S8, S9, and S11
TSR 33	The operations in the TRANSACT architecture shall be in near-real-time or real-time.	
TSR 34	When the system is online, the turnaround time issuing a request to obtain advisory response will not be more than 1 minute.	
TSR 35	The image registration component in the architecture shall be able to align pre-operative CT images with 2D X-ray images and return the result within 5 seconds.	

TSR 38	To detect software failures and aging in the architecture, it shall be necessary to constantly monitor the availability of services, nodes, and the status of software containers and/or virtual machines (when used)	S4, S6, S13, S15
TSR 41	The state-of the art resource provisioning strategies in the architecture should be generally intended to rely on behavior modelling and prediction to achieve higher accuracy and shorter computation times.	S7, S8, S9, S11, S14
TSR 47	The architecture should support Latency-critical applications. The latency between observed traffic by the roadside camera and the autonomous vehicle should be below 500 milliseconds.	S6 used together with S7, S9, and S11 (and, when needed, S1 and S2)
TSR 50	The latency of the platform for transferring 3D reconstructions back to the console / viewing workstation shall not exceed 10 seconds. The latency of the platform for the transfer of images and calibration data to the reconstruction platform shall not exceed 20 seconds.	S6 used together with S7, S9, and S11 (and, when needed, S1 and S2)
TSR 53	If used in the Use Case the Edge Servers in the architecture should provide real-time or non-real-time control for the devices. Real-time tasks may be vital which involve human safety. Therefore, it is vital to have a reliable system which reacts when it is needed and how it is needed. The physical reliability requirements for Edge servers providing services is similar to Cloud Computing	S9
TSR 54	To adapt the architecture to cope with the dynamic events dynamic data transfer rate, important event detected by sensors, and network connectivity, the architecture should support the reconfiguration of the data flow with the aim of balancing the load between the edge nodes, or to redirect the data flow to the node with the best conditions (resource availability and lower response latency).	S1, S2, S7, S8, S9, S14
TSR 55	The components in the architecture should be able to monitor events, resources and errors.	S6, S13
TSR 56	The architecture should support the monitoring of the data transmission rate of the devices.	S16
TSR 58	The TRANSACT system architecture should support error detection.	S6, S7, S13
TSR SP-1	The system shall ensure data integrity across all processing steps in the solution, including to ensure the integrity of disparate data originating from multiple sources and/or multiple locations in the device-edge-continuum.	S5
TSR SP-2	The system shall allow operation in different operational modes and ensure safe transition between those modes to support system reconfiguration and fall-back solutions in case cloud-based functionality is not accessible or not performing properly.	S1, S2, S11
TSR SP-3	Applications running on the system shall be scalable in terms of their use of system resources.	S3, S11, S15, S17

TSR SP-4	The system shall allow modular safety assurance for device versus edge versus cloud parts and shall support independent releasing of (updates to) device versus edge versus cloud parts.	S12

Table 4: Technical safety and performance requirements (TSRs) addressed by D3.3

1.3 Structure of this deliverable

The structure of this deliverable is as follows. We first provide an overview on the 3-layer categorization of the solutions provided in this deliverable in Section 2. To maximize harmonization between the concepts developed in D3.1 and the solutions being developed in D3.3, we have adopted the same categorization. Most of the solutions provided here are also a direct realization of the concepts introduced in D3.1. Sections 3, 4, and 5, introduce application-related, cross-cutting, and platform-related solutions developed by the partners of the task, respectively. Section 6 discusses the relations and tensions between the solutions for safety and performance and solutions for security and privacy. Finally, we conclude this deliverable in Section 7.

2 Overview of selected solutions for safety and performance

2.1 Introduction

Aligned with the concepts and requirements formed in Deliverable D3.1, we have created an overall structure to organise the selected solutions. These solutions are the means to implementing a selection of relevant concepts referred to in D3.1. This section presents this brief overview of the solutions and discusses the alignment between the D3.1 and the solutions being developed in D3.3. We have in total, three categories of solutions and 17 distinct solutions provided by the partners.

2.2 Overview of selected safety and performance concepts

Similar to D3.1, we categorize the solutions being developed in our task to three categories:

- **Application solutions**: they target the *applications*. An application is the functionality that implements a particular solution/technique/method to help the end-user to perform a specific task. It 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.
- **Cross-cutting solutions**: they 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.
- **Platform solutions**: they target the *platform*. The 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, network, hypervisors, operating system, containers, cloud computing services and run-time libraries.

This categorization allows system designers to distinguish the platform (the infrastructure of a system) from the applications that run on this platform or are enabled by this platform. Any solution that considers both the status of the application and the platform into account would then be categorized as a cross-cutting solution. The selected solution classes, based on these three categories, are shown in Figure 1.

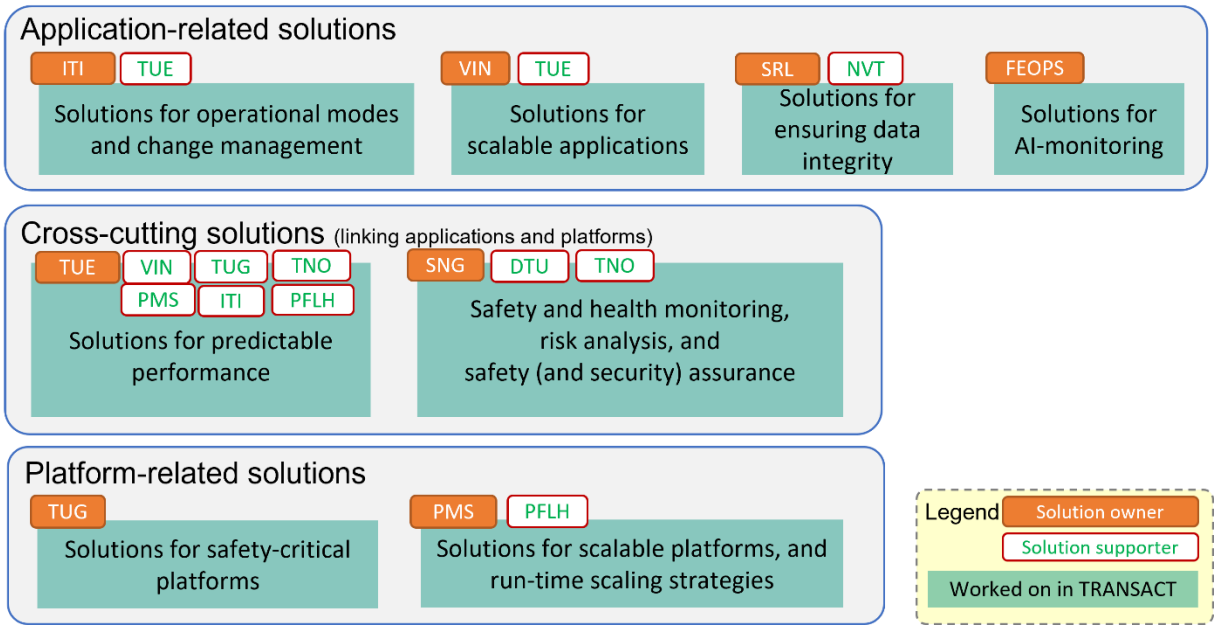


Figure 1: Selected safety and performance solutions

A solution class may contain a small number of related solutions as needed to cater to the needs of various use cases and their domain characteristics. In the next sections, the selected solutions are described per solution category (Application, Cross-cutting, Platform) and class (for example, predictable performance, safety and risk analysis, or AI monitoring).

2.3 Alignment with D3.1

As explained earlier, this deliverable provides solutions to achieve or realize the concepts and requirements established in the Deliverable D3.1. However, as for some of these concepts and requirements there exist mature solutions or technologies to achieve them, not all concepts introduced in Figure 8 of the Deliverable D3.1 required a specific solution to be designed by the TRANSACT project partners. For example, “Concepts for application service level agreements” are already realized (implemented) by commercial cloud providers.

We have combined the concepts for “safety and risk analysis” and “health and safety monitoring” together with the concept for “continuous safety and security assurance” (from D3.1) into the “Solutions for safety and health monitoring, risk analysis, and safety (and security) assurance”. The reason is that with the enormous efforts of the TRANSACT partners (in particular, SNG together with all use case owners), we succeeded to tackle these concepts and requirements via the integrated risk-analysis and safety and security monitoring tools developed by SNG. Namely, the developed tools and solutions (presented in Section 4.3.1 and 4.3.2, respectively) provide an integrated environment for risk analysis (before and after transforming the systems to the device-edge-cloud continuum), monitoring safety and security properties/requirements of the system, and providing safety and security assurance following the standards defined for each use case domain. All use cases UC1, UC2, UC3, UC4, and UC5, are involved in this process and trying to apply the same framework on their use cases.

3 Application solutions for safety and performance

3.1 TRANSACT project harmonised needs and expectations

To align with the main objectives of the TRANSACT project, this deliverable is in charge of exploring various solutions to enable safety-critical CPSs to safely transfer into edge and/or cloud platforms. For some use cases, this includes moving real-time and/or safety-critical functionalities (tasks) to edge or cloud platforms and for some other use cases, this means supporting added-value and AI functions in the device-edge-cloud continuum.

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 safety is always warranted. This requires applications to offer various operating modes, be scalable, and support seamless change of operation modes. The solutions on “Mode change management on the device” (Section 3.2.1) and “Mode change coordination” (Section 3.2.2) specifically address these needs and provide mode-change protocols, manager, and coordinator for the safety-critical applications to support the fall-back mechanisms and allow the system to safely change its mode of operation.

One of the key enablers of the operational mode management and change transition is the scalability of applications, namely, applications should be able to provide a trade-off between their Quality-of-Service (QoS) level and their resource requirement. This ability allows a mode manager (in all layers of the TRANSACT reference architecture) to decide whether an application should be executed on the cloud or should fall back to edge (or device) in case the cloud resources (processors, memory, etc.) become temporarily unavailable to the user. Section 3.3 “Solutions for scalable applications” discusses how such trade-off can be made by introducing an architecture framework for the applications (which is being applied on UC1 as a proof of concept).

Furthermore, as indicated by the use cases of the TRANSACT project, supporting value-added AI-services on the cloud is one of the major needs of CPS systems. However, unlike deterministic, non-AI software where post-deployment monitoring receives less attention, the AI-services require continuous monitoring (to support continuous learning, investigate and correct their mistakes, and improve the quality of their outputs over time). Section 3.4 “Solutions for AI Monitoring” presents an initial attempt on a customizable, plug-and-play monitoring tool for AI applications.

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. Section 3.5 “Solutions for ensuring data integrity” investigates solutions to provide data quality monitoring and data quality improvement during the lifecycle of an application that run on the cloud.

The selected application solutions are highlighted in Figure 2.

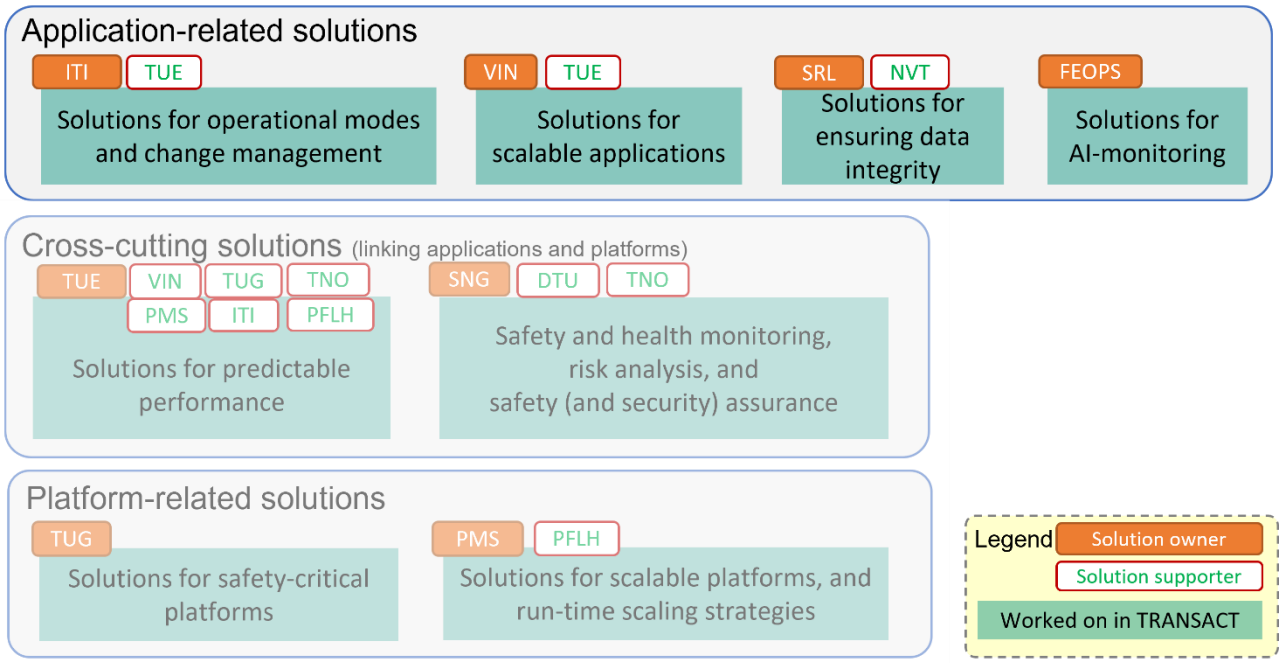


Figure 2: Selected application solutions

3.2 Solutions for operational modes and change management

3.2.1 Mode change management on the device

3.2.1.1 Overview

In this section, a solution for managing multi-mode real-time applications executing on devices is presented. This refines the concepts for “operational mode run-time management” and “mode change techniques” included in the “concepts for operational modes and changes transition” in deliverable D3.1. As mentioned in that deliverable, multi-mode systems exhibit multiple behaviours implemented using operational modes, each comprising its own set of tasks (or threads) with different characteristics (periodicity, execution time, criticality...). The transition between two modes is initiated by a Mode Change Request (MCR) and involves finalizing the execution of the set of tasks belonging to the old mode and starting the set of tasks of the entering new mode. This transition needs to be managed following a Mode Change Protocol (MCP), which defines the allowed transitions, the type of transitions, the offset times and other conditions and constraints to guarantee the timing requirements of the system.

The presented solution fits within the *operational mode manager* in the device-tier of the TRANSACT reference architecture, which is the core component in charge of performing mode changes in the device. In the same way, as discussed in section 5.5.5 of deliverable D3.1, the related concepts are associated with the same core component.

This deliverable presents a generic solution that is designed as a middleware library, and focuses on describing the necessary interfaces and mechanisms to handle mode switching.

3.2.1.2 State of the art

The proposed solution, that implements the *operational mode manager*, focuses on integrating an MCP into a real-time application. The challenge is to manage the MCRs and the transitions between modes with the

aim of guaranteeing the feasibility of the system. Up to now, the protocols to schedule multi-mode real-time systems have been profoundly studied from different approaches. Real et al. (Real, 2004) describe a survey of MCPs for single-processor, fixed-priority, preemptively scheduled real-time systems, and propose several new protocols along with their corresponding schedulability analysis and configuration methods. In the field of the multi-processor platforms, several articles have described solutions to address this problem. Yomsi et al. (Yomsi, 2010) propose two transition protocols to schedule multi-mode applications on uniform multi-processor platforms. Also, V. Nelis et al. (Nelis, 2009) consider the global and preemptive scheduling problem over identical multi-processor system. With respect to heterogeneous multiprocessor hardware, Goossens et al. (Goossens, 2019) present a multi-mode protocol for mode-dependent tasks on reconfigurable hardware. This approach aims to handle predictable switches between different tasks sets and different hardware settings.

Regarding the implementation of those MCPs into a real platform, the number of studies found is smaller. Sáez et al. (Sáez, 2012) propose a framework for multi-processor, multi-mode real-time applications. They model the different system modes and transitions using UML finite state machines (FSM), and present a code generation tool to generate a basic implementation of the defined model in Ada language. Additionally, T. Chen et al. (Chen, 2018) provide SafeMC, a specification language to design and evaluate MCPs. They define a set of fundamental primitives that most MCPs tend to be built on and demonstrate the utility of this language with an implementation of different MCPs into the Xen hypervisor.

The expected solution must provide the capability of configuring different MCPs, as those enumerated previously, based on the idea described on SafeMC (Chen, 2018). Additionally, the approach should be flexible like the framework presented by Sáez (Sáez, 2012). As a result, during the course of the TRANSACT project, a generic *operational mode manager* is being developed, with the intention of being used as a reusable core component.

3.2.1.3 Innovation step

As concluded on the related work, during the design phase of a multi-mode device application, the correct MCP must be selected and configured to ensure its successful execution. In order to facilitate the later development, a generic *operational mode manager* should be implemented, that allows configuring different protocols to manage the mode switching on the device. The proposed solution has been designed as a multi-platform middleware with the aim of implementing different MCPs, depending on the requirements of the application. This generic multi-platform middleware will allow reusing this core component in different systems based on the TRANSACT architecture.

The solution for managing mode changes on the device requires:

- To **define the operational** mode model that includes:
  - *operational modes* that can be executed,
  - *allowed transitions* between modes,
  - *set of tasks* that are enabled in each mode, and
  - *temporal attributes* of each task depending on the selected mode.
- To **configure the MCP** that comprises both the actions to be performed, depending on the type of tasks during a mode transition, and the guards to be applied to these transitions, usually represented as offsets to wait before releasing each task (defined by (Chen, 2018)). The type of tasks during the transitions are:
  - *old tasks*, that are active only in the previous mode,
  - *new tasks*, that are active in the new mode,
  - *unchanged tasks*, that are active in both modes and maintain the same configuration,
  - *changed tasks*, that are active in both modes but its configuration does not stay the same.

- To **receive MCRs** from external agents or from the own device application.
- To **process the received MCRs** and apply the corresponding changes to the task configuration based on the selected MCP and on the mode transition.

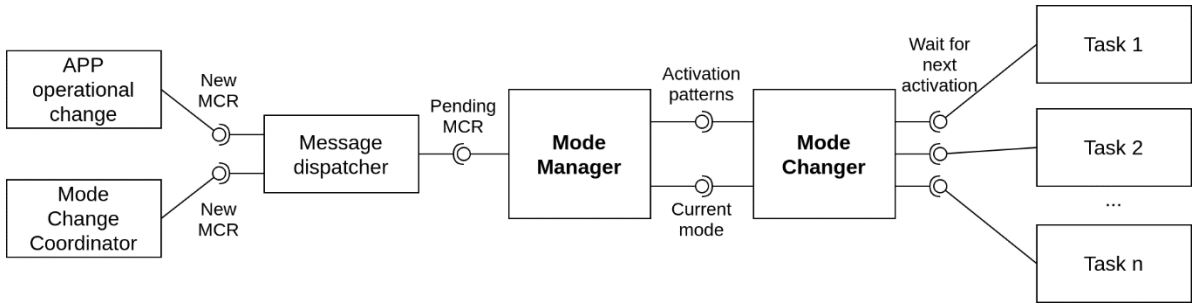


Figure 3: Operational mode manager interfaces

To meet with these requirements, the proposed solution must offer two main interfaces (Figure 3) to interact with the application on the device and with other external agents, such as the operational mode coordinator on the edge (see section 3.2.2). It comprises both components, mode manager and mode changer, as depicted in Figure 3. *Mode manager* is in charge of processing new MCRs and updating the activation patterns of the different tasks. After that, *mode changer* controls the transitions between modes, performing the mode changes by releasing the waiting tasks. Regarding the external interfaces, the first one is proposed to accept MCRs that are stored in the message dispatcher, pending to be processed by the *mode manager*. Additionally, the other external interface provided by the *mode changer* is a directive that is called from the **periodic tasks** (or threads) configured in the real-time application on the device. This directive, called *wait_for_next_activation*, blocks the execution of the calling tasks until its next activation is performed. The activation will be released depending on the selected mode and the configuration of the calling task in that mode.

As a remark, not all the tasks in a real-time application depend on a periodic activation, such as those that depend on the completion of another task or on the activation of an asynchronous event. These other tasks will follow the regular method used previously: semaphores, messages, etc.

The main function of the *operational mode manager* is no other than to block the activation of periodic tasks, coordinating their execution based on the selected operational mode. It is not intended to replace the OS scheduler, so the actual activation of the tasks will continue to be handled by it.

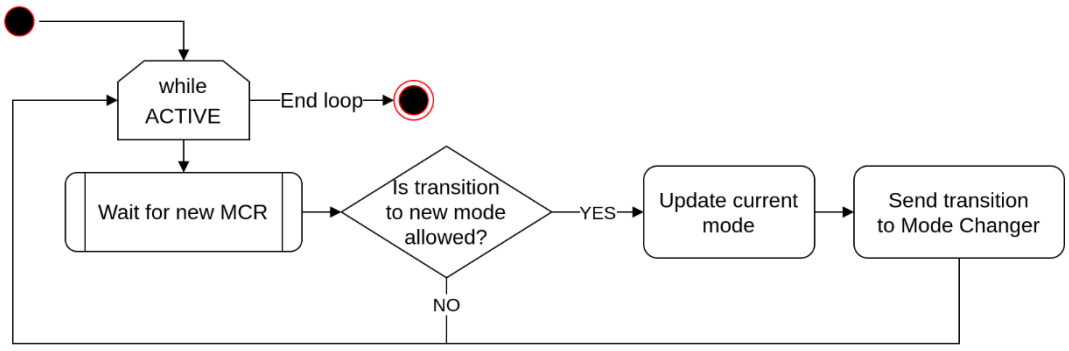


Figure 4: Mode Manager execution flow

The main procedure of the *operational mode manager* comprises three routines:

- **Process pending MCRs**: In this routine (Figure 4), M*ode Manager* reads from the *message dispatcher* queue the MCRs that are processing pending. If the MCR can be performed, since it is a transition allowed by the model, the status of the different tasks is updated, recalculating the next activation if necessary. These calculations are made considering the defined MCP and its configuration.
- **Send task releases**: The second routine is executed on the M*ode Changer* (Figure 5). A task, which calls the *wait_for_next_activation* directive, and which is included in the tasks-set of the current mode, should be released when its next activation time is due. This routine is responsible for verifying if any tasks in the current tasks-set has expired its activation time and wake-up it. Finally, it must remain asleep until the time for the next activation of one of the tasks expires or a new MCR arrives.

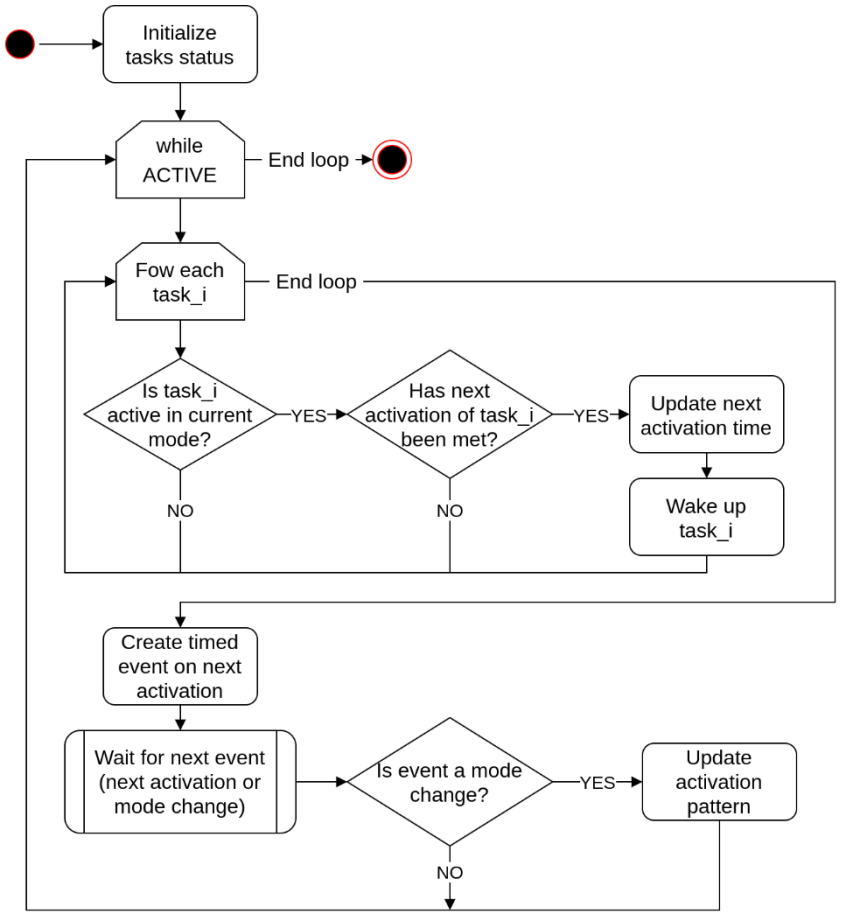


Figure 5: Mode Changer execution flow

Regarding the implementation of the *operational mode manager*, the different functionalities have been encapsulated as components (Figure 6), facilitating its validation and reusability. At the top of the component model, the *Mode Manager* component can be found, which executes the main routine defined in Figure 4. Concurrently, the routine described in Figure 5 is executed in the *Mode Changer* component.

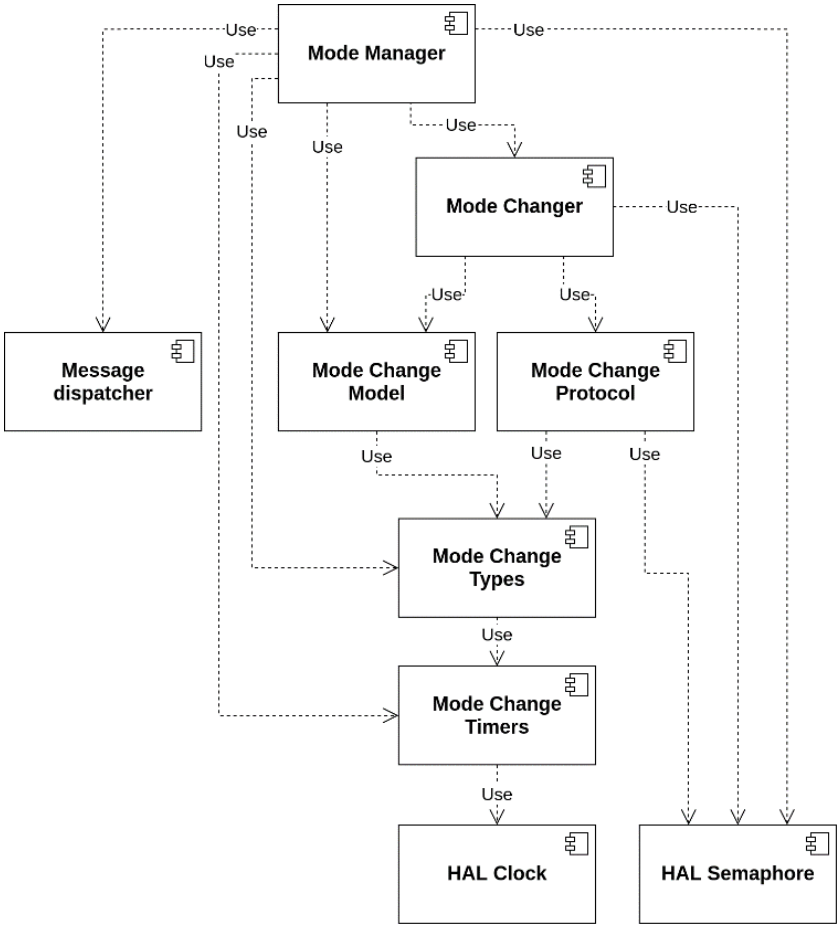


Figure 6: Operational mode manager software components

On the other hand, there are a *Mode Change Protocol* and a *Mode Change Model, where* the protocol defines the actions to be performed depending on the transition that is executed and the model describes the modes, the transitions allowed, and the tasks and configuration associated with each mode. In addition, the *HAL layer* components abstract calls to the system to facilitate the migration of the solution into different operating systems and platforms. These components must be the only ones that depend on the operating system. Finally, the *Message dispatcher* component encapsulates a queue where MCR messages are stored until they are processed by the *Mode Manager* routine.

The described solution should be used as a framework for the development of the operational mode manager service of the TRANSACT reference architecture. This development, which begins as a proof of concept (PoC), needs to be extended and improved to:

- Include most of the directives described in SafeMC (Chen, 2018) that allow the service to configure multiple MCPs.
- Evaluate the migration of the component in different operating systems.
- Validate its operation by integrating it into a complete system.

3.2.1.4 Application to use case(s)

The solution proposed in this section implements the core component of the *operational mode manager*, defined in the TRANSACT reference architecture. This solution is intended to be integrated and validated in the horizontal demonstrator, but it has been designed to be integrated into other demonstrators if necessary.

Therefore, it could be used for any device within the TRANSACT use cases, if it needs to manage several operational modes, either functional modes or degraded modes.

As indicated in deliverable D3.1, this solution could be integrated into UC4. If, for some reason, the response time of the image registration process in this use case cannot be met, for example, due to network unavailability or to limited bandwidth, the system should recognize this situation and switch to another operational mode to mitigate the problem. The management of the mode changes in the device could be performed by the operational mode manager described in this solution.

In the first stage of the design of this solution, a PoC has been developed that includes a multi-mode application with several tasks or threads. This development is intended to generate evidences to validate the operation of the proposed solution.

The initial version, that has been implemented in this PoC, includes three type of guard transitions, as a subset of the options defined by (Chen, 2018):

- **Immediate**: The new job of the task must be released immediately.
- **Offset from MCR**: An offset, relative to the MCR instant, defines the delay to wait before next activation.
- **Offset from LR**: An offset, relative to the last release of the task, defines the delay to wait before next activation.

The application to be validated runs over Linux (with PREEMPT-RT patch), and configures the scheduler in SCHED FIFO mode, allowing different priorities to be allocated for each thread. The selected platform is an Ubuntu Server running in a virtual emulator with a dual-core processor, trying to define an environment similar to that of the device.

The multi-mode application comprises three modes of operation and its transitions, as defined in Figure 7. The normal operation mode performs critical and mission tasks, to meet requirements of high functional performance. In case of need, a degraded mode can be executed, maintaining only critical tasks, and reducing the frequency of the tasks that allow it. The third mode is the reconfiguration mode, which introduces a new task in charge of the management of the application settings. The Mode Manager main routine is implemented as the highest priority task with the aim of activating the different tasks in the application. The configuration summary of the different tasks is detailed in Table 5.

Task name	CPU affinity	Priority	Period		
			Normal	Reconfiguration	Degraded
Mission 1	0	3	200 ms	-	-
Soft critical	0	4	100 ms	200 ms	200 ms
High critical	0	5 (higher)	50 ms	50 ms	50 ms
Reconfiguration 1	0	2	-	200 ms	-
Mission 2	1	1	150 ms	-	-
Reconfiguration 2	1	0 (lower)	-	150 ms	-

Table 5: Multi-mode application - task configuration

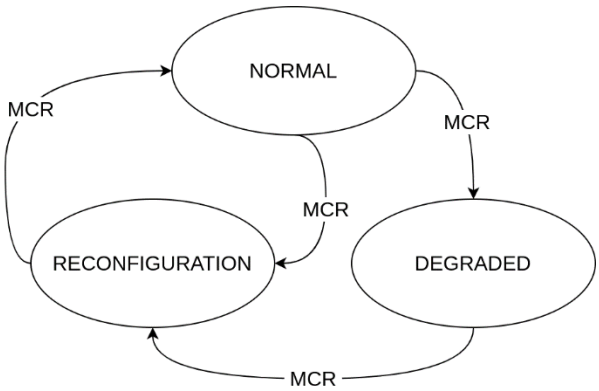


Figure 7: Operational mode manager PoC: Operational modes

To complete the configuration of the operational mode manager, the actions described in the MCP must be defined:

- Jobs of old tasks will finalize its execution if its activation was released before the MCR.
- New tasks will start the execution as soon as possible after the MCR.
- Changed tasks will start the execution after an offset from the last release equals to its period in the new mode.
- Unchanged tasks will maintain its periodicity from its last release.

The timing behaviour of the system under test is monitored using *LTTNG* library and the results are depicted and analysed using *Eclipse Trace Compass*. These open-source applications facilitate the extraction of useful performance information, which has been captured during the execution of the system under test. Figure 8 shows the thread flow of one execution of the system, including MCR to evaluate the transitions to the different modes.

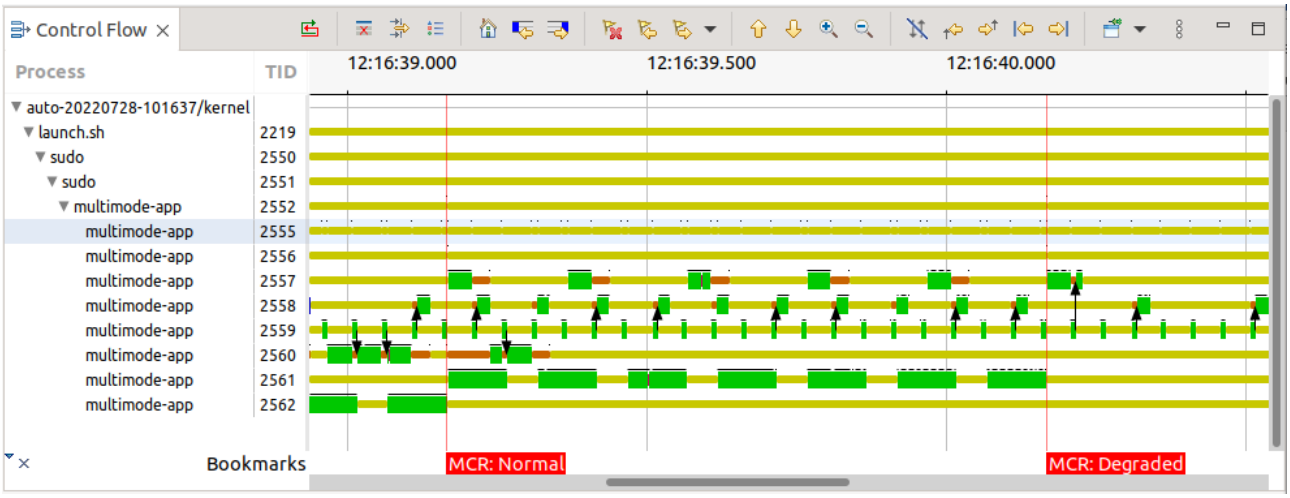


Figure 8: Operational mode manager PoC: Thread flow

From these results the following conclusions can be observed:

- *LTTNG* and *Eclipse Trace Compass* made easy to validate the correctness of the solution and observe the PoC behaviour, ensuring that it is running as expected.

- The task-set for each mode is executed as configured, and the rules and offsets defined in the MCP are met at all transitions.
- The Mode Manager task is executed when a new MCR arrives.
- The Mode Changer task runs only if a waiting task needs to be released. Otherwise, it is sleeping, reducing the jitter introduced.
- The priorities defined in the configuration are met.

It is intended to improve this solution during the development of the TRANSACT project, and present the final results in the second version of this deliverable. The focus of the effort will be on:

- Increasing the number of guard transitions in the configuration of the MCP
- Migrating the proposed solution to an RTOS
- Validating the core component in the horizontal demonstrator.

3.2.2 Mode change coordination

3.2.2.1 Overview

In this section, a solution to synchronise multi-mode real-time distributed applications is presented as a complement to the solution presented in section 3.2.1. This refines the concepts for “operational modes and change transitions” in deliverable D3.1. Additionally, it is related to the concepts for "performance modelling and prediction" and "performance management" included in the "concepts for predictable performance". Although it is not detailed in that deliverable, it could also be associated with other concepts aligned with the adaptability of the application, such as the "concepts for safe and secure modular updates" or the "concepts for health and safety monitoring".

As mentioned in deliverable D3.1, in case a requirement, or a Service Level Objective (SLO), is violated, or is predicted to be, the application should update its behaviour to avoid or mitigate the violation and ensure the correct execution of the system, or at least, guarantee a minimum QoS. As explained in section 3.2.1, this adaptive behaviour can be modelled using a multi-mode system, which defines the different configurations that will be executed depending on the desired behaviour. In the case of distributed systems, the difficulty of implementing this type of applications increases. This is due to the necessary synchronisation between the different nodes during the transition from one operational mode to the next. It is at this point that a coordination mechanism becomes essential.

The presented solution fits within the *operational mode coordinator* in the edge and cloud tiers of the TRANSACT reference architecture, which is the core component in charge of propagating and synchronizing the operational mode changes in the system. In the same way, as discussed in section 5.5.5 of deliverable D3.1, the related concepts are associated with the same core component.

This deliverable presents a generic solution that is designed as a middleware library, and focuses on describing the necessary interfaces and mechanisms to handle synchronization of multi-mode applications.

3.2.2.2 State of the art

One of the main characteristics of cyber-physical systems is their ability to respond and adapt to changes in the environment. This not only includes the capability to mitigate malfunctions, but also to modify normal modes of operation due to mission needs. In recent years, many studies have focused on the design of adaptive systems and the description of their architectural patterns. Many of them have been developed from the MAPE-K loop, already described in deliverable D3.1, as the base architecture to enable self-

adaptation of these systems. Consistent with the TRANSACT reference architecture, *operational mode coordinator* in TRANSACT is concerned with the Plan and Execute phases of the MAPE-K loop.

Different classifications of key properties that are associated with self-adaptive system have been proposed by Andersson et al. (Andersson, 2009). The purpose of this study is to establish a baseline from which these key aspects can be easily identified and compared. Regarding the architectural patterns, Weyns et al. (Weyns, 2013) present a selection of MAPE-K patterns that model different types of interacting MAPE loops with different degrees of decentralization. On the side of decentralized systems, Weißbach et al. (Weißbach, 2017) define an approach to perform multiple related adaptations of a distributed software system without the need for a central coordinator. Also, Arcaini et al. (Arcaini, 2015) describe a conceptual and methodological framework for formal modelling, validating, and verifying distributed self-adaptive systems, exploiting the concept of Abstract State Machines to specify decentralized adaptation control by using MAPE computations. On the other hand, Sáez et al. (Sáez, 2012) propose an approach using finite state machines (FSM) to describe operational modes and transitions, and a framework of real-time utilities that implement the required behaviour in a centralized approach from the planner point of view.

Despite the solutions offered to facilitate the coordination of distributed systems, some proposals have been encountered in the IoT field. For example, the FIWARE framework (Cirillo, 2019) offers an open-source IoT platform that focuses on data context management, which could be used to homogenise the provided data from the heterogeneous devices. In the same way, OpenIoT (Kim, 2014) proposes an open service framework for IoT to coordinate and adapt the services of the different layers in a distributed system, from device to edge and cloud. These solutions focus on the coordination of heterogeneous device networks from the point of view of the context management. They are useful to adapt the data provided by the different devices, but they do not serve as a complete solution for self-adaptation.

From the annotated bibliography, different proposals have been observed to coordinate and synchronise the operational modes of the system, most of them based on self-adaptive systems. In contrast, no solutions have been found at a mature product stage that can be easily integrated into the TRANSACT architecture as a core service. However, during the course of the project, the search for existing solutions that can fit within the *operational mode coordinator* service will continue.

3.2.2.3 Innovation step

In the current state of the art, solutions have been found that allow mode changes to be propagated and synchronised through a distributed system, in order to adapt the behaviour of the application. Unfortunately, the ones that could meet the requirements are not in the product stage, which means that they cannot be easily integrated and maintained within the TRANSACT solution. For this reason, we have considered to develop a new solution that fits the requirements of the operational mode coordinator service. This solution must be based on the developments described in the state of the art, but adapting to the specific requirements of a TRANSACT core solution, such as reusability or scalability.

To begin to understand the need for a solution to coordinate mode changes, Figure 9 defines an isolated device from the perspective of the MAPE loop model. The monitoring (M), analysis (A) and planning (P) phases are developed from within the application, which is responsible for deciding if a mode change is necessary. For example, by monitoring (M) the temporal behaviour of the different tasks, it is possible to analyse (A) that the performance of the application has decreased. The established plan (P) defines that in order to not compromise the operation of critical tasks, the application must be executed in degraded mode, so it requests this change to the executor (E) using an MCR. In this case, the execution (E) phase matches with the core component of the *operational mode manager* (see section 3.2.1) that manages application mode switching.

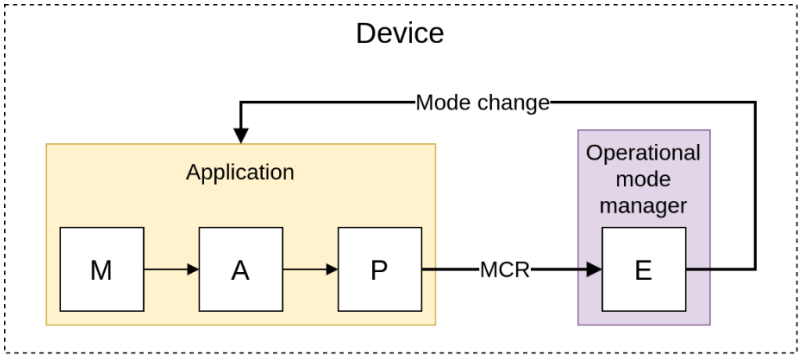


Figure 9: Isolated MAPE device

The complexity increases when the device is no longer isolated and instead belongs to a distributed architecture (Figure 10). Mode change requests can now come from a service running on the edge, this means that the device must be able to process MCRs from the outside and return the result of that request. If the requested change is not allowed by the model that defines the transitions between modes, the device must inform the edge that it has been denied.

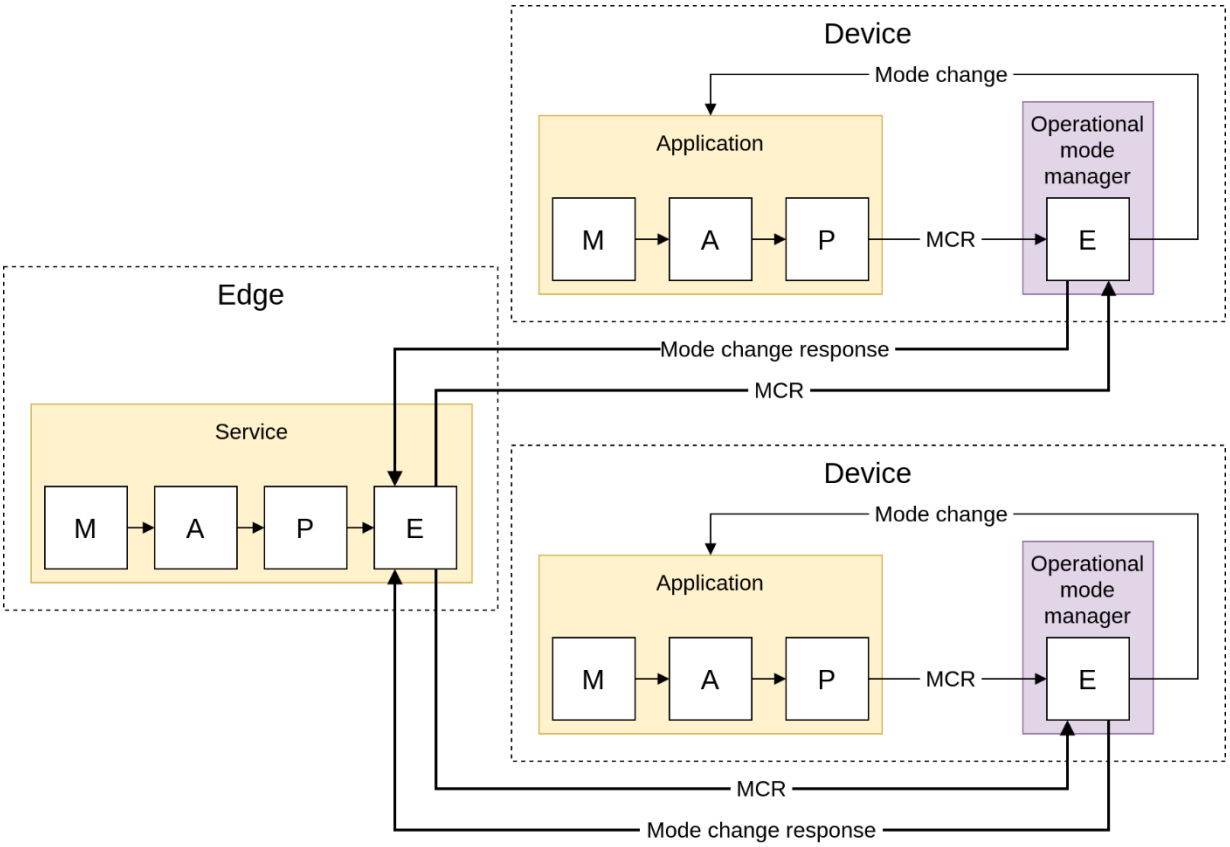


Figure 10: MAPE device into edge architecture

In this case, the executor at the edge is in charge of sending MCRs to the different devices. The method to communicate with each type of device, as well as the device discovery service, are not part of this solution. Another service within the TRANSACT architecture will have to take care of these aspects.

As discussed in the previous section, the *operational mode coordinator* runs at the edge or cloud tiers (Figure 11). This service comprises both the planner and executor phases of the MAPE loop. The monitoring and

analysis phases are performed by other services that detect changes in the system behaviour. The planner receives these changes as events, that are used to update the future behaviour of the system.

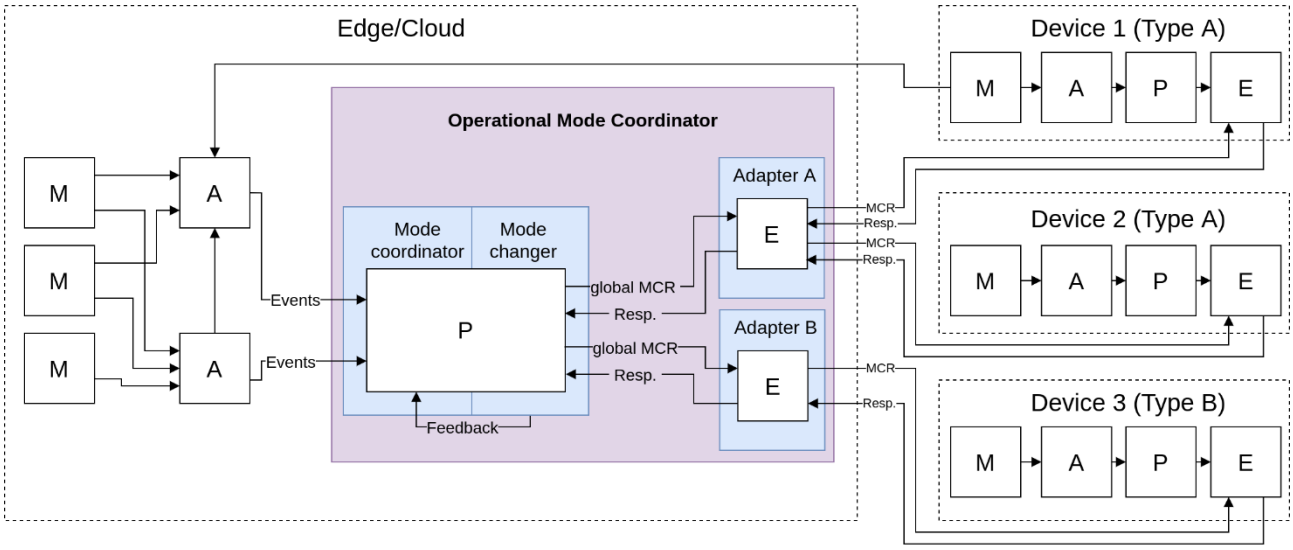


Figure 11: Operational mode coordinator on the Edge

The planner stage in the *operational mode coordinator* is composed of the following components:

- ***Mode Coordinator***, which is in charge of processing the incoming events, provided by the analysis services. This component can be designed as an FSM, that defines the different global states of the system and the transitions between them. The global states do not have to match with the states of each of the devices, as long as global states represent the behaviour of the complete system and states of a single device defines a partial actuation. Regarding the implementation of this component, the specific behaviour of this FSM is application-dependent and defined as a configuration, but the logic to implement it must be designed as a common middleware.
- ***Mode Changer***, which is responsible for propagating the mode change requests in the system. This component generates *global MCRs* that are translated by the adapters to configure each device to the desired mode. Once the change has been made, or in case of failure during the transition, the *mode changer* must send the feedback of the operation to the *mode coordinator* so that it can act accordingly.

The executor phase is performed by the *adapters*, that are the components in charge of translating the *global MCRs* into the specific *MCRs* expected by each device. One adapter is responsible of one device or a group of devices of the same type. It sends the MCR to the device and wait for the response to ensure the transition can be executed.

The described solution should be used as a framework for the development of the *operational mode coordinator* service of the TRANSACT reference architecture. This development, which begins as a design concept, must eventually be implemented during the course of the TRASANCT project. For this, the following concepts must be considered:

- Although no solutions have been found at the time of writing this deliverable, existing ones should be studied in search of one that meets the requirements of the operational mode coordinator.
- The scalability of the system must be considered, keeping in mind that a consistent global state of the system is a priority concept.

3.2.2.4 Application to use case(s)

The solution proposed in this section implements the core component of the *operational mode coordinator*, defined in the TRANSACT reference architecture. This solution is intended to be integrated and validated in the horizontal demonstrator, but it will be developed so that it can be integrated into other demonstrators if necessary.

The final component will be presented in the following version of this deliverable, as evidence that the solution works, integrated into an application based on the TRANSACT architecture. This application must configure different operational modes at the system level, such as a normal, degraded or low power operational mode. Several devices must be connected to the edge through their respective adapters, which will translate the global modes into specific modes of each device following a previously configured mapping.

The behaviour of this PoC must try to emulate the behaviour of a critical multi-mode system, trying to reproduce possible failures in the devices or at the edge to check the robustness of the solution.

3.3 Solutions for scalable applications

3.3.1 Overview

Scalability of applications is one of the key enablers for operational mode management and change transition. 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.

In Figure 12, 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.

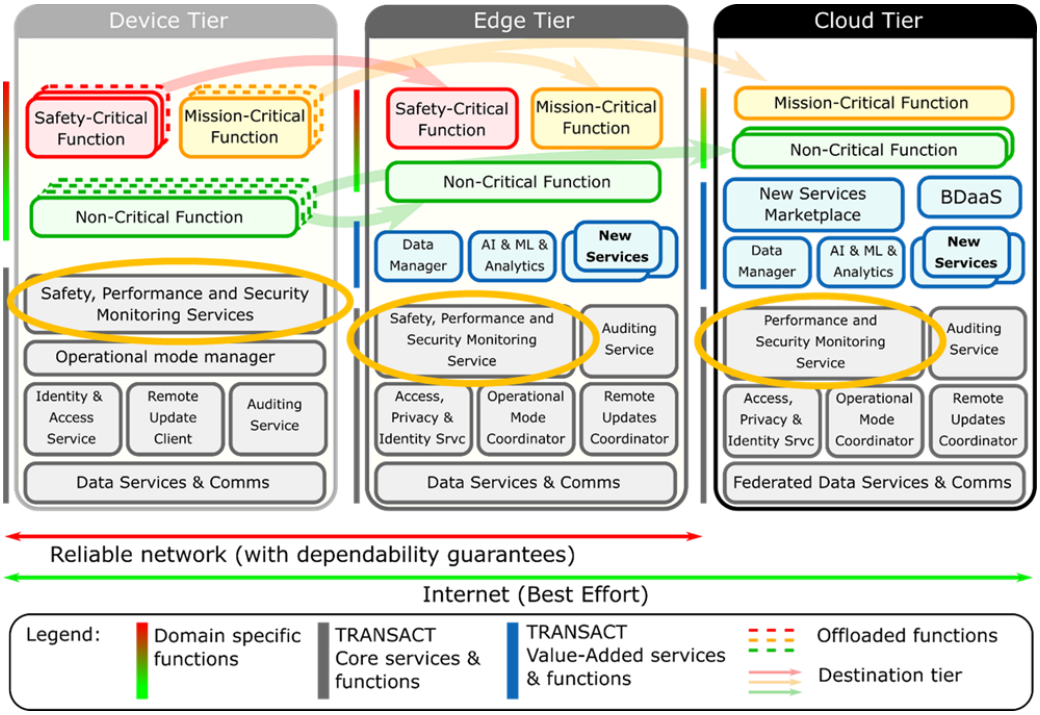


Figure 12: Scheduling of scalable applications in the TRANSACT reference architecture

3.3.2 State of the art

There is extensive work done for Secure and Scalable Quality of Service for Critical Applications (Wyss, 2021). Most of this work is related to distributed IoT Systems to deal with unreliability of network resources. For many critical applications, reliability, security, and scalability of communication systems are of paramount importance. Critical applications often rely on leased lines, sacrificing on scalability and flexibility while incurring high cost. On the other hand, best-effort Internet connectivity cannot satisfy the reliability and security requirements.

For TRANSACT we want to address scalability of applications by considering a broader set of resources. The network can be considered as a resource that is needed for the application to execute, but also memory budget, number of computations and data-bus bandwidth are important resources where the execution of applications depend on. Hence, we want to allow scalability of applications by exchanging the Quality of Service (QoS) by an application with any type of relevant resources.

3.3.3 Innovation step

As explained in Deliverable D3.1 we introduce an architecture framework to address scalability of applications. We introduce the following framework elements:

1. A set of relevant resources and an algorithm to predict the resource allocation by the application.
2. A set of operations points of the application, each with different defined resource budget.
3. A set of QoS levels, each representing different quality levels of the application in terms of accuracy of the results, availability of the system, latency of the response or throughput of the tasks to be performed.
4. A scheduler that selects the operations point, thus a certain resource budget, depending on the predicted resource allocation.

3.3.4 Application to use case(s)

For 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.

As a proof of concept, we use the road-side camera sensor with edge device of pedestrian, bicycle and vehicle detection as a part of the application where we want to introduce the scalability of the application.

We plan to realize the scalability with the above-introduced framework by implementing the framework elements as follows:

1. Set of relevant resources: memory, operations per second.
2. Set of operations points: we define a set of framerates and resolutions for AI processing and object tracking. Many other options exist such as the number of neural network layers, the maximum number of objects to be tracked, the type of tracking algorithm, enable/disabling parts of the algorithm such as image stabilization, etc. For now, we limit the operations point by only selecting different framerates and resolutions.
3. Set of QoS levels: Each operation point represents a different quality level. For a lower framerate, the latency of the system increases which will reduce the maximum speed of the vehicle. Lowering the resolution will reduce the distance for detecting, e.g., pedestrians. This reduces the safety or limits the routes for the remote vehicle.
4. The scheduler: we will implement metrics to measure the amount of occupied memory and the CPU and GPU load by measuring the execution time for processing a set of video frames. Based on the time budget (40ms at 25 frames per second) we set a threshold to dynamically change the operation mode.

3.4 Solutions for AI-monitoring

3.4.1 Overview

This section provides an overview of the presented solution for AI-monitoring. As discussed in Deliverable 3.1, Section 5.8 “Concepts for AI monitoring”, being able to monitor deployed applications incorporating AI is essential to the applications life-cycle. This is a fundamental difference with deterministic, non-AI software where post-deployment monitoring receives less attention. Although several solutions exist to monitor AI deployments, these often seem to lack the flexibility of being able to be used as an add-on to existing software and need to be built into these applications. While this could be an option for new software, this is often not ideal e.g., if this requires an update of a medical device which can take a fair amount of time to be approved. In addition, these solutions often require specific data formats mainly focusing on classification tasks and machine learning tasks with simple data, such as age or class distribution, but do not allow experts to monitor complex data, such as annotated DICOM images. The solution described below presents an initial attempt on a customizable, plug-and-play monitoring tool for AI applications.

In Figure 13, the orange ellipse shows the components in three tiers of the TRANSACT reference architecture where the AI-monitoring tool can take place. The solution described below is mainly related to the “Performance and Security Monitoring Service” in the cloud tier but is extendable to the edge and device tier, depending on the application.

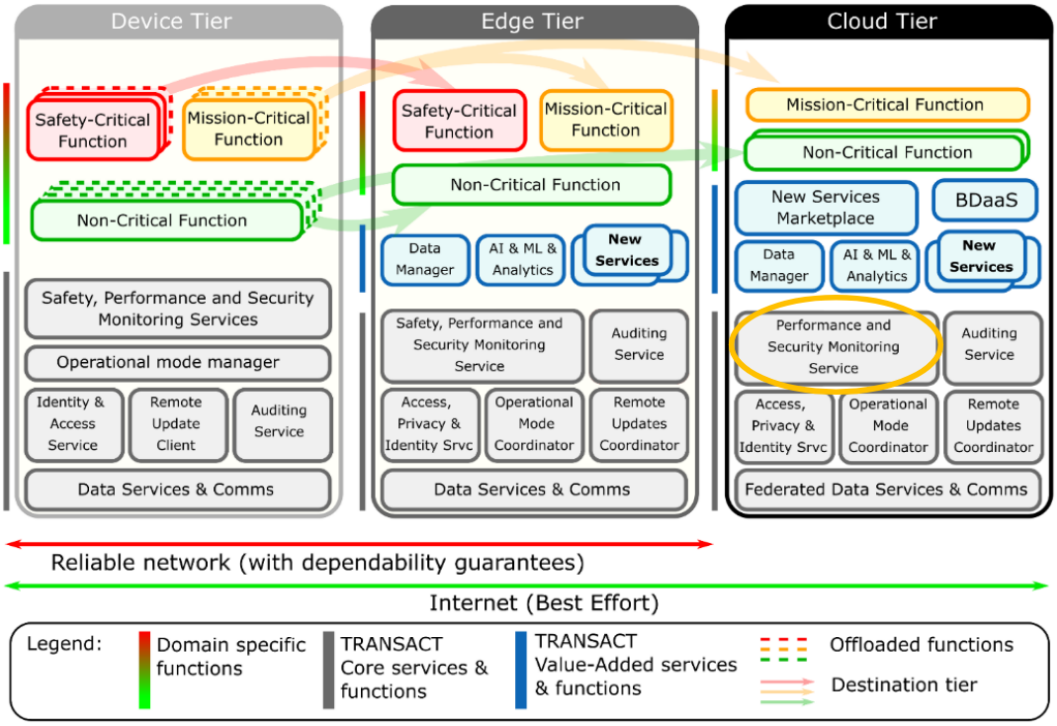


Figure 13: Solutions for AI-monitoring in the TRANSACT reference architecture

3.4.2 State of the art

Interest in explainable AI and AI-monitoring has significantly increased over recent years, partially thanks to the increasing adoption of AI research into production software (e.g. Siri, FaceApp, Salesforce Einstein, etc.). Although add-on packages provide basic functionalities to monitor simple data, these still lack high customizability, require predefined data formats and often need to be integrated into the existing software to unlock the live monitoring features EvidentlyAI (Evidently AI, 2022), Great Expectations (Great expectations, 2022) and DeepQu (Deequ, 2022). Adding new custom metrics or visualizations for custom data is not supported by default which requires the user to dig into the internal specifications of these codebases. Additionally, these tools are mainly focused on simple classification tasks for machine learning applications, while the field of AI entails much more, including deep learning with many more high-dimensional features which can be difficult to monitor. Also the possibility for AI monitoring using AI algorithms is absent.

Finally, monitored metrics can be divided into 2 main groups: Operational metrics and ML-specific metrics. While the existing tools mainly focus on the latter, an AI-monitoring tool capable of e.g., combining both GPU utilization with request time and accuracy could provide useful insights as well.

In conclusion, for the TRANSACT prototype we are taking the first steps towards an AI-monitoring toolbox that has a modular architecture, making it easy to add new analysers, metrics, visualizations and update existing ones. These tools do not have to be integrated into already existing software but can be an add-on, live updated dashboard which makes this highly useful in systems where many small releases are not possible (e.g., in medical devices).

3.4.3 Innovation step

Our AI-monitoring tool requires a data generation, a data storage framework and a data monitoring toolbox. Our solution allows any data in any storage format according to business requirements. This can be completely new data or already generated data available in production. The single requirement should be

that the data can be read by external software (e.g., through REST API calls). Figure 14 visualizes the solution in a possible real-world application: Prior to any AI application, a research & development team builds the AI models and application. Once deployed, data is generated by these AI models and stored into a data storage. Optionally, the output can be reviewed/corrected by a human-in-the-loop to obtain feedback early on. The AI monitoring toolbox can be used to continuously monitor the performance of the application and possibly trigger any model adjustments to maintain high quality output.

The main component in the solution is the AI monitoring toolbox. Here, the stored data is fetched using custom fetchers, analysed using custom analysers, compared and exported also using custom objects, while still maintaining a modular design which fits the high versatile architecture design inside TRANSACT. Where current AI-monitoring tools fail to allow customizable metrics from any storage location, our solution succeeds in giving the user control on what and how this application data should be monitored.

The monitoring toolbox architecture is split up into 4 main concepts, also depicted in Figure 14. In this figure, the data generation and data storage can be any system (already in-place or newly developed), while the highly customizable AI monitoring component is an extension to the application. Components of this figure are:

1. Analysers: This component reads data from the storage, given filters, sorting parameters, (custom) metrics, etc. It allows the software to create its own live feed from the data, which can then be used by either a dashboard or exporter to visualize or write reports, respectively.
2. Metrics: These are the core of the monitoring tool as these define the performance, data drift, target drift, explainability, Gaussian analysis, etc. of the AI models being monitored. Any metric can be added to the toolbox, including other AI based algorithms to evaluate the AI predictions.
3. Exporters: Current systems tend to use more and more data which could slow down a live updated feed without much gain. Exporters allow users to generate custom reports that can be stored and read at any given time. These provide a screenshot of the state of the AI application at the given moment in time and can be useful for e.g., quarterly reporting of the application performance.
4. Dashboard: If users would like a live feed of the current status of the AI application, the dashboard component can help to provide insights using visualization tools. It contains generic plotting components, using the attributes from analysers and metrics to populate a web UI.

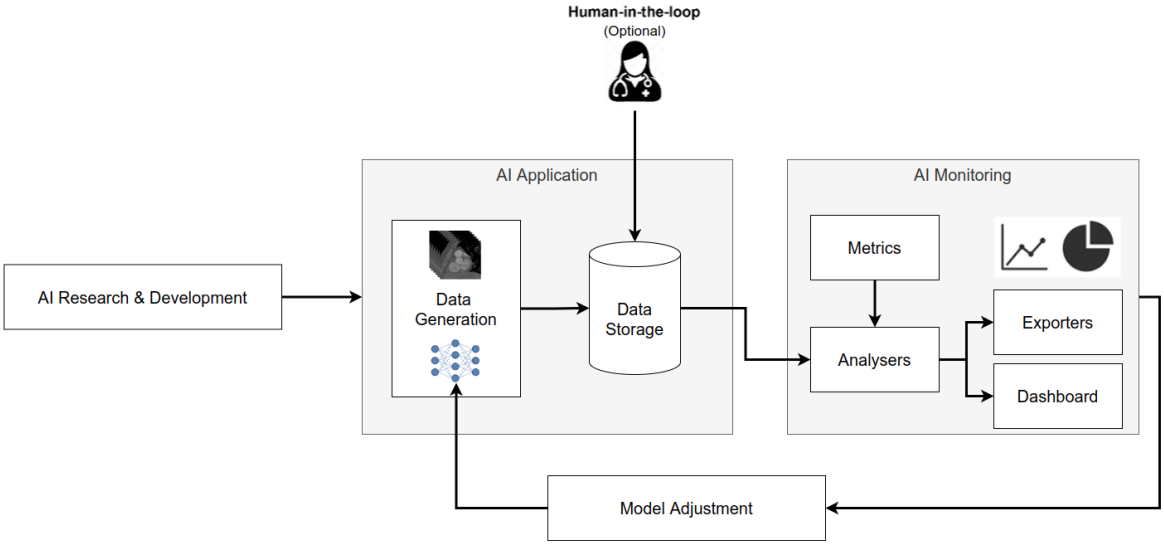


Figure 14: The proposed AI-monitoring architecture solution.

3.4.4 Application to use case(s)

All use cases collect information to monitor and act upon based on this received data. The AI monitoring architecture is very modular which allows the user to add data from any structured data format such as databases, csv files, structured hard drive folders, etc.

As described in Deliverable 3.1, an ideal use case for the provided solution is the medical use case (UC4). AI models can be monitored to detect any anomalies or changes in performance. As a reaction to model degradation, the monitoring can inform the user and even trigger alternative scenarios such as automated retraining or sending notifications through dedicated channels (e.g., e-mail, Slack, Microsoft Teams, SMS, etc.).

Additionally, in the case of UC 1 and UC 2, one could monitor the discrepancy between the real-world data and the predicted digital twin data or the predicted navigation routes and actually chosen ones. Thresholds can be used to notify users of any alarming anomalies.

Figure 15 depicts a mockup of the analyzer output, visualized by a dashboard. Different navigation pages allow the user to view different visualization pages. An example of a page is shown: This could contain some information on the specified AI project or general application, some useful trends annotated with green or red arrows depicting improvements and degradations respectively of requested metrics. Additional plots can pinpoint moments in time when a threshold was surpassed, as indicated by the red vertical line on the middle plot, or any in-or outgoing data distributions, as depicted in the two lower graphs. A second page could provide a more detailed card of specific input data (e.g., should this sample be signaled as an anomaly), while a third page could provide functionality to compare different metrics and detect correlations between them.

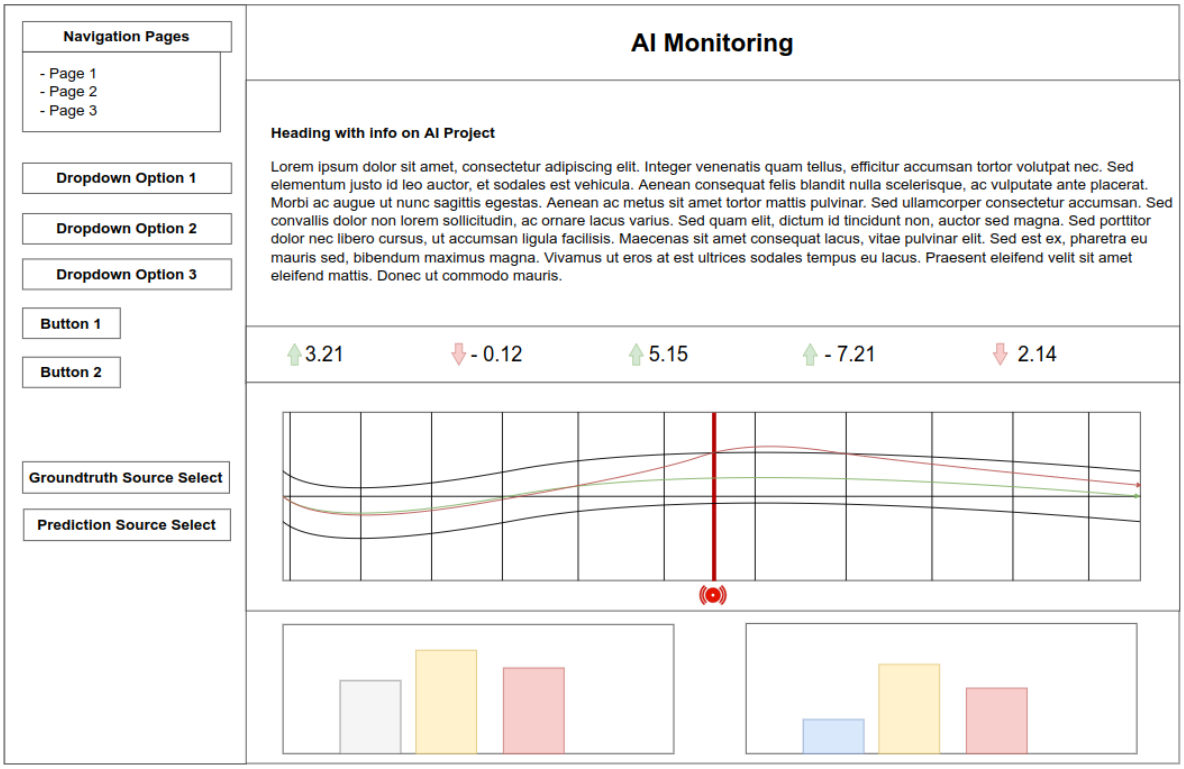


Figure 15: Mock-up of the AI monitoring dashboard web UI, providing visualizations for the monitored application

3.5 Solutions for ensuring data integrity

3.5.1 Overview

This section describes an overview of the solution for ensuring data integrity. Data integrity is the key element of any ML application, as such applications are built on data. Data integrity refers to accuracy and consistency of data throughout its lifecycle. Examples of some data inconsistencies include missing data, data out of range, type mismatch. However, data integrity is difficult to maintain. This is due to the complexity of data and its processing pipelines. For example, data is often transformed during the stages of data collection and pre-processing to ensure its proper fit for use. Such transformations need to be applied on all the same type of data throughout the pipeline and across multiple pipelines, as it is common that ML models receive input data from multiple pipelines. One example are ML models consuming data from streaming and historical (batch) sources. Additionally, ML applications often have multiple versions of ML models deployed, which complicates the process of monitoring data integrity across multiple models/versions. However, without a proper solution for monitoring and ensuring data integrity in ML applications across their whole lifecycle, the performance of ML applications/models may be significantly degraded because of data errors and inconsistencies.

In Figure 16, the purple ellipses show the components in three tiers of the TRANSACT reference architecture that are concerned with ensuring data integrity.

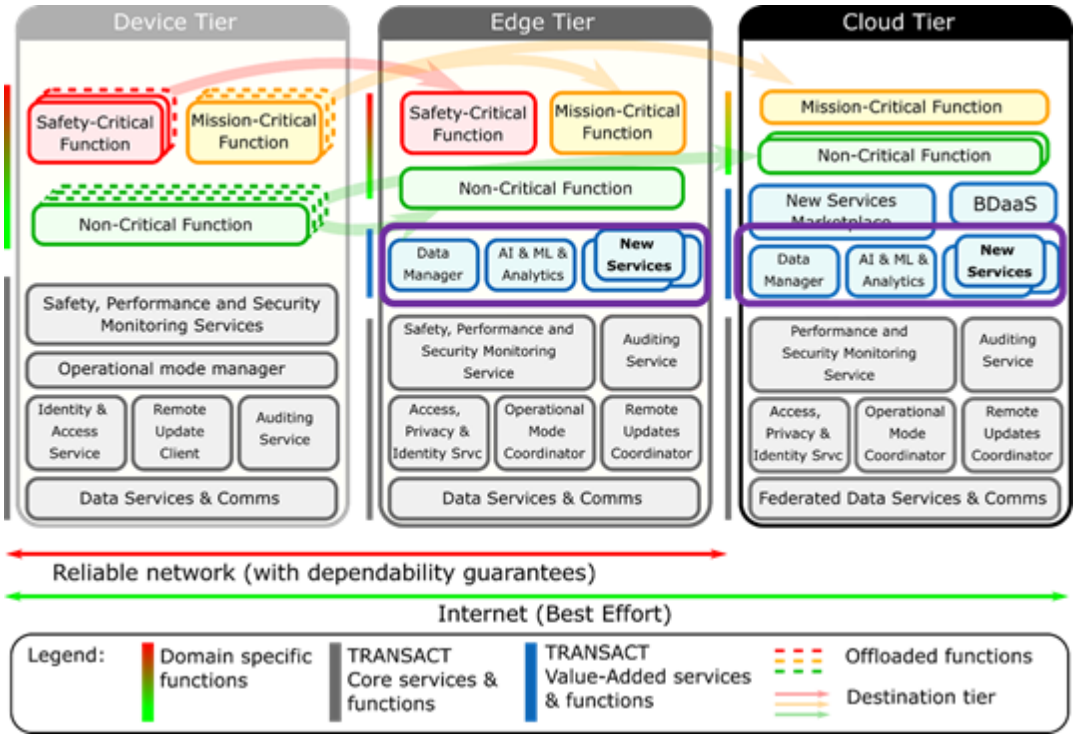


Figure 16: Scope of the relevance of the solution for ensuring data integrity in the TRANSACT reference architecture

There are specific types of data inconsistencies in different use cases. In UC2, for example, additional problems arise due to manual input of vessel data, inaccuracy in International Maritime Organization (IMO) ship identification number and MMSI data (Maritime Mobile Service Identity) as well as missing historical relationship between them. Moreover, registered AIS (Automatic identification system) positions sometimes

"jump" between different vessels and regions. In order to guarantee the quality of the TRANSACT AI-based solution during the whole lifecycle, most types of quality issues should be detected and cured.

3.5.2 State of the art

One possible solution for detecting data inconsistency issues is implementing data quality monitoring. Such approach can detect mean shift in multi-attribute data (Yu, Wu, & Tsung, 2019) as one type of data inconsistency. Another possible solution for detecting data inconsistency issues is implementing change-point detection for real-time monitoring high-dimensional streaming data (Zhang & Mei, 2020). For ML-based applications, 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. Some examples of the metrics in the maritime domain, in the context of UC2, include the *frequency of Automatic Identification System (AIS) data reception* and *data noise.* The first metric enables tracking of vessels in maritime navigation, thus improving navigational safety. The second metric denotes different types of AIS data inconsistency, for example AIS data can be duplicated and it can contain noise such as incorrect timestamp, speed over ground, position readings. This is sometimes the results of dense traffic causing AIS message collision.

Another recent solution for improving data inconsistency issues involves explainable AI. This approach assumes that by providing explanations for the causes of data quality gaps we can more efficiently improve data quality. However, existing solutions for explainable AI monitoring are mostly focusing on simple metrics, such as accuracy and feature drift, lacking the ability to monitor more complex metrics, such as data inconsistency and quality, or bias.

3.5.3 Innovation step

Our approach to ensuring data integrity throughout its ML lifecycle involves two steps: *Data quality monitoring* and *Data quality improvement*.

In the first step, we apply explainable AI monitoring, with the goal to provide explanations for the causes of data inconsistency. For example, in the context of UC2, we monitor the frequency of Automatic Identification System (AIS) data reception metric. In maritime, AIS messages are used to track vessels, thus improving navigational safety. AIS data is collected via base station receivers (on land) and via satellite receivers. Monitoring the frequency of AIS data reception can notify that there is a missing AIS reception for a given period of time, within a longer interval in which valid AIS data were recorded. By checking with other data sources, we analyse and provide an explanation of the cause of the missing reception.

In the second step, we attempt to improve the quality of inconsistent data. Depending on the type of data inconsistency, this process 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.

3.5.4 Application to use case(s)

This solution can be applied to the maritime use case (UC2). UC2 deals with developing reliable ML-assisted 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) of the target ML models, measuring a set of metrics. After the ML models are deployed in production, streaming data keeps feeding into the ML pipeline, going through data quality monitoring and data quality improvement processes.

In UC2, the solution will monitor input AIS data in order to detect and categorize bad quality data automatically. Based on quality category, specific treatment will fix data issues respectively.

More complicated quality issues are related to ML-assisted routing services. The solution will collect AI – specific quality metrics. Such metrics will indicate signs of degradation of AI-models, needs for calibration and tuning, etc. For example, metrics based on changes in AIS data distribution will be used as a trigger for models re-training and re-calibration. Another important metric to trigger a process of re-training of AI-models will be based on a monitoring of use of route advices and feedbacks from navigators. In the beginning, the AI model will produce good routing advices because it is newly trained and proposes quite efficient and safe routes. Due to the good quality of the advices, the routes produced by the AI model will be often accepted by users. After some time, however, the model and quality of proposed routes start to degrade. Navigators will reject advices (or give a negative feedback) more often due to results inefficiency or lack of safety awareness.

The solution will monitor such metrics and control quality of both AIS data used by AI-models and routes produced by AI models.

4 Cross-cutting solutions for safety and performance

4.1 TRANSACT project harmonised needs and expectations

Safe operation of safety-critical distributed CPSs that have been deployed on a distributed device-edge-cloud continuum requires both design-time and run-time techniques to model, analyse, and monitor performance of these systems as well as their safety (and security) properties. Moreover, there is a need for proactive resource management for the resources on the network, cloud, and edge platforms with the goal of guaranteeing certain timing properties (e.g., for real-time applications running on the cloud) or improving performance and resource efficiency (for real-time and non-real-time applications). This section discusses two broad set of solutions developed or extended by the TRANSACT partners on "predictable performance" and "safety and health monitoring, risk analysis, and safety (and security) assurance", where in the former category, the focus is mostly on performance aspects and in the latter category on the safety (and security) aspects.

In Section 4.2, we explore three sub-categories of performance related solutions focusing on "performance monitoring", "performance modelling and predication", and "performance management". The goal of the first category (Section 4.2.1) is to monitor a system's performance on the device-edge-cloud continuum. Partners of the TRANSACT project had joined force to perform a survey on the existing monitoring techniques and technologies. In Sections 4.2.2.1, 4.2.2.2, and 4.2.2.3, partners have investigated solutions for performance prediction and analysis using AI-based, simulation-based, and formal-method techniques and discussed how they can be used by different use cases of the TRANSACT project. Section 4.2.2.4 provides a performance assessment solution for workflow simulation of hospitals (related to the UC4). Finally, in Section 4.2.3.1, partners introduce a scenario-based performance management and reconfiguration technique that allows adjusting system resources to the applications and adapting their mode at runtime while preserving some properties about the system performance.

Section 4.3 presents solutions for risk analysis and risk management (Section 4.3.1), machine-learning based real-time monitoring techniques for detecting safety and security anomalies (Section 4.3.2), and a set of techniques and solutions for mapping and scheduling applications on the network (Section 4.3.3). Finally, in Section 0, partners investigate monitoring techniques for the continuity of services on the cloud, addressing topics such as availability of cloud platform resources over time.

The selected cross-cutting solutions are highlighted in Figure 17.

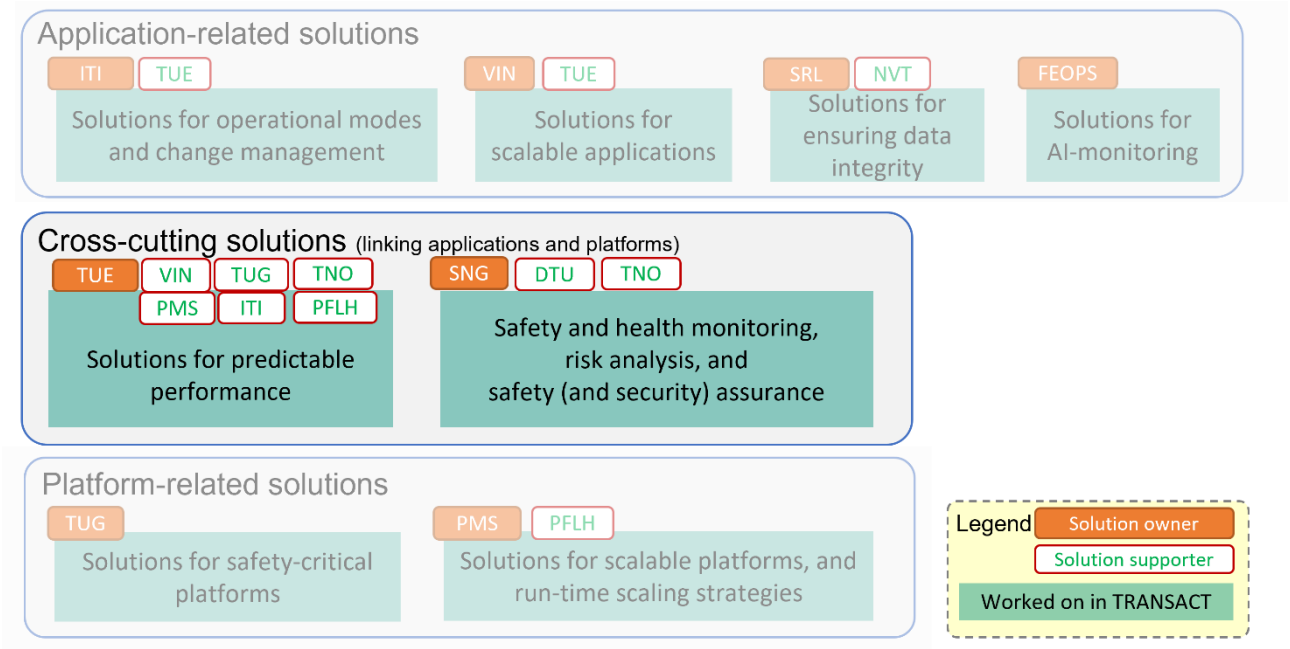


Figure 17: Selected cross-cutting solutions

4.2 Solutions for predictable performance

4.2.1 Performance monitoring

4.2.1.1 Performance observability and monitoring

4.2.1.1.1 Overview

Observability is provided through three essential data types, abbreviated M.L.T. (Goniwada, 2022) (Li, et al., 2021):

1. **Metrics** provide time-based numerical measurements, time-series, on elements of the application or system. Application-level metrics depend on the nature of the particular application, but examples include the number of users, the application throughput, or the number of successful payments. Examples of relevant system-level metrics include usage of resources, such as CPU, memory, and network bandwidth.
2. **Logging** collects application-generated structured or unstructured text, e.g., a developer logs information about what happens in the application during its execution.
3. **Traces** provide an understanding of how requests flow between functions, components, or microservices in a system over time.

The three data types are complementary and provide insight into application health and performance (Goniwada, 2022). For example, a trace can tell you which part of a service is slow, but metrics and logs are needed to explain why. Best practices involve making sure that all collected telemetry (M.L.T.) data is correlated, such that they can be combined and analysed together, and stored in a common persistent storage to create a single source of truth for what is happening in the system (Goniwada, 2022).

Having good system observability and monitoring telemetry data in a system has many benefits. It is essential to facilitate effective fault analysis and debugging (Zhou, et al., 2021). It also helps create an understanding

of the run-time architecture of a system, e.g., through dynamic inference of service dependencies, which is particularly helpful in cloud applications where services can be dynamically instantiated and short-lived (Sigelman, et al., 2010) (Li, et al., 2021). In the context of performance, it enables tracking actual performance against specified performance budgets, to pinpoint performance bottlenecks, and to determine the root cause of a performance anomaly (Sigelman, et al., 2010) (Li, et al., 2021). It is also a prerequisite for model learning and calibration, auto-scaling, and to create resilience through mechanisms like rate regulation and circuit breaking (Heinrich R. , et al., 2017).

In terms of the TRANSACT reference architecture, most of the infrastructure related to observability and monitoring is a part of the Safety, Performance, and Monitoring Services across the three tiers, as shown in Figure 18. However, virtually all other parts of the reference architecture can be impacted, depending on which application or system services are made observable and how (further described below). The Safety, Performance, and Monitoring services also interact with the Operational mode managers/coordinators to trigger mode changes based on monitored data.

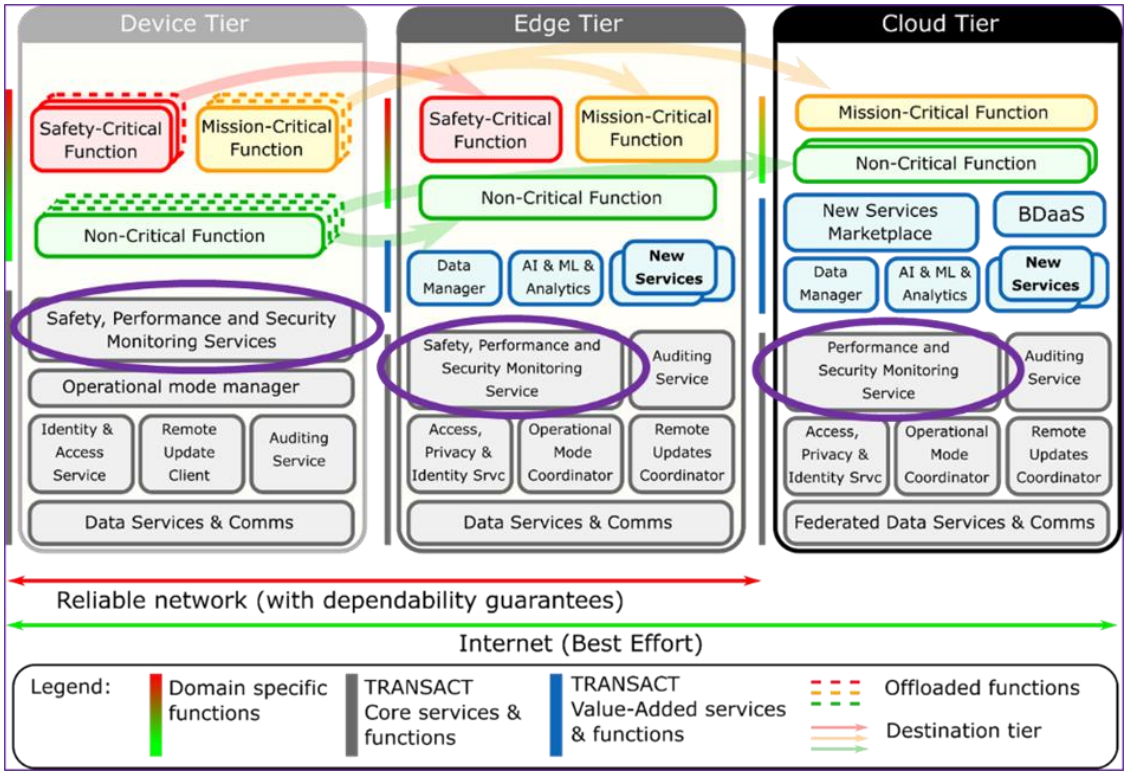


Figure 18: Observability and monitoring is a part of the Safety, Performance, and Monitoring Services in the TRANSACT reference architecture.

4.2.1.1.2 State of the art

There are many open and commercial state-of-the-art tools that enable observability and monitoring of telemetry data. We will proceed by introducing a few of these tools, with a focus on well-established opensource tools with large communities, as these are freely accessible and have a large impact on the community. These tools have largely been developed and open sourced by Big Tech companies during the past decade, with the primary goal being to enable observability and monitoring in large-scale microservice architectures running in public or private cloud environments. This hence corresponds to the cloud tier of the TRANSACT reference architecture, and possibly the edge tier, depending on what this looks like. However, there is no fundamental technical reason why they could not also be applied in all tiers, as microservice

architectures and containerization are getting increasingly prevalent also in edge environments (Li, Lemieux, Gao, Zhao, & Han, 2019). However, there are also mature tools with origins in the embedded domain. We will start by reviewing the state-of-the-art originating from the cloud tier and then continue with solutions for embedded systems in the device tier.

State-of-the-art: Cloud Tools for Metrics

There are several tools open-source tools for monitoring and visualizing metrics, such as Graphite (Graphite, 2022) and Prometheus (Prometheus, 2022). We will briefly introduce the latter, as it is both more recent and feature-rich. Prometheus is a fully integrated time-series database management and monitoring system. It was originally developed at SoundCloud but joined the Cloud Native Computing Foundation (CNCF) in 2016. It has a large user community and is the de-facto standard tool for collection and monitoring of metrics. The typical use of Prometheus includes configuring a number of endpoints, i.e., physical/virtual machines or containers, containing relevant system or application metrics and a time stamp, along with a period indicating how often the metrics should be updated. The Prometheus server will then scrape these endpoints with the desired period and store the metrics in memory and on local disk. For ephemeral (short-lived) service instances with lifetimes too short to scrape, a push gateway is provided. Each Prometheus server is stand alone and does not rely on other storage. However, it can be combined with external time-series databases like Mimir (Grafana Mimir, 2022), if highly scalable long-term multi-tenant storage is required. Prometheus comes with an alert manager that allows rules to be configured that trigger alerts, e.g., based on threshold values for different metrics, and notify users through a variety of notification clients, e.g., for e-mail, Slack, or SMS. Scraped metrics can be queried using a query language, called PromQL, or visualized using a web interface. The user interfaces of both Graphite and Prometheus are very simple, and they essentially rely on other tools to provide a good user experience. Both tools can be connected as data sources in Grafana (Grafana, 2022), which provides a better user interface with powerful data visualization, queries, and alerts across multiple data sources.

State-of-the-art: Cloud Tools for Distributed Tracing

Similar to metrics, there are also several popular open-source tools for distributed tracing. Two prominent examples include Zipkin (Zipkin, 2022) and Jaeger (Jaeger, 2022), both inspired by Google Dapper (Sigelman, et al., 2010). These tools are quite similar and follow a generic five stage tracing and analysis pipeline (Li, et al., 2021). We briefly describe Jaeger, the more recent and feature-rich open-source option. Jaeger was released by Uber Technologies in 2015 and was open sourced in 2017, before graduating as a top-level project of the CNCF in 2019.

The Jaeger architecture includes an agent, residing on a host or in a container, that collects spans from services on the host or in the container (see Figure 19). Spans are the basic unit of work in a trace tree that essentially tracks the start and end of the execution of a service, as well as the id of the parent service that called it. The Jaeger agent pushes all collected spans to a centralized collector that stores them in a database. This trace database can then be queried based on the time covered by the trace, its duration, and involved services from a user interface. Jaeger supports multiple deployment options, as it can be deployed either as an all-in-one binary in a single process, or as a scalable distributed system. The key analysis features of Jaeger include providing a timeline view of the spans in a trace, similarly to the view on the right-hand side of Figure 19. This makes it possible to see where time is spent in the execution of the application, which allows the scope of problem-solving activities to be narrowed down. It is also possible to generate dependency graphs that show which services are communicating with each other across a set of traces, helping users understand the (run-time) architecture of the system. Lastly, it is possible to compare sets of traces, e.g., traces capturing successful and failed executions, and visualize the differences in (the order of) service invocations. All of these analysis techniques are frequently used in industry practice (Li, et al., 2021).

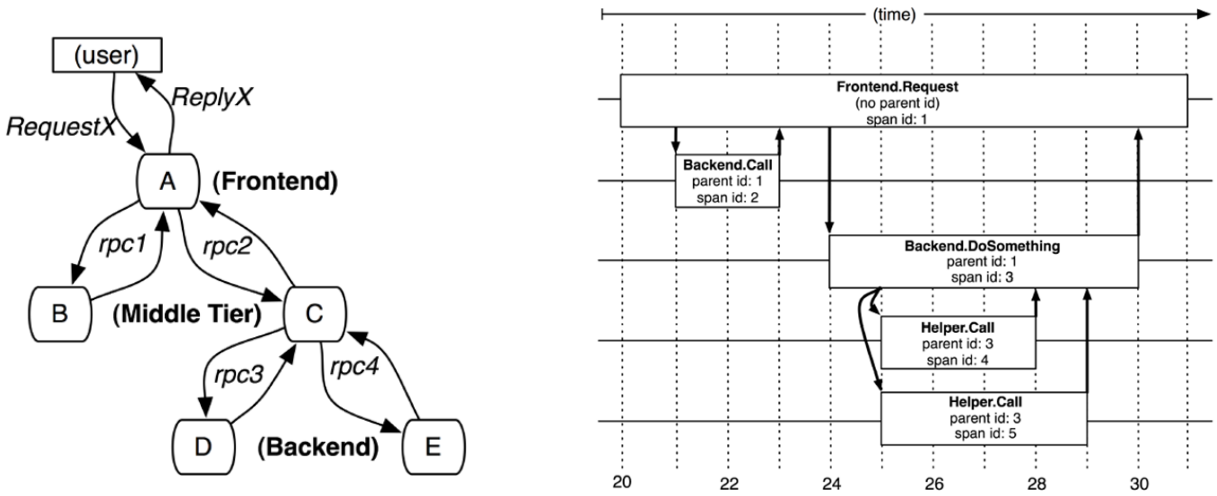


Figure 19: A trace tree indicating a set of communicating services (left), along with a corresponding timeline view of its spans (right) (Sigelman, et al., 2010).

Instrumentation

Instrumentation is the practice of using client libraries in applications or infrastructure to generate telemetry data. Many of the tools mentioned above, e.g., Zipkin, Graphite, Prometheus, provide their own client libraries in a variety of programming languages. However, these libraries are often limited to the particular tool, causing a lock-in. With the proliferation of tools to collect, visualize, and analyse telemetry data, there has been a push to create and consolidate technology-agnostic instrumentation. For example, OpenTracing (OpenTracing, 2022) was created by Uber in 2016 and OpenCensus (OpenCensus, 2022) by Google in 2018 with the goal of providing vendor-neutral instrumentation. These tools were merged into OpenTelemetry (OpenTelemetry, 2022) in 2019. Currently, OpenTelemetry is an incubating project with CNCF. OpenTelemetry is capable of creating and collecting metrics, traces, as well as logs and can be used in combination with many of tools mentioned above, e.g., Prometheus, Jaeger, and Zipkin, through a variety of exporters.

Sidecar Proxies and Service Mesh

Having good vendor-agnostic tools for instrumentation and creation of telemetry data is only helpful if they are consistently applied by developers. However, some developers may forget to instrument their code, fail to do this consistently, or even choose not to for concerns about its impact on performance (Sigelman, et al., 2010). This may result in incomplete traces that fail to capture the complete journey of requests through the system, making problems with the system harder to resolve. This problem is often addressed by instrumenting middleware libraries, e.g., for communication, to ensure that all remote procedure calls are traced (Sigelman, et al., 2010) (Li, et al., 2021). However, the emergence of new microservice technology, like sidecar proxies and service meshes enables cleaner software architectures that better separate applications (business logic) and platform infrastructure (Jamshidi, Pahl, Mendonça, Lewis, & Tilkov, 2018). This is illustrated in Figure 20.

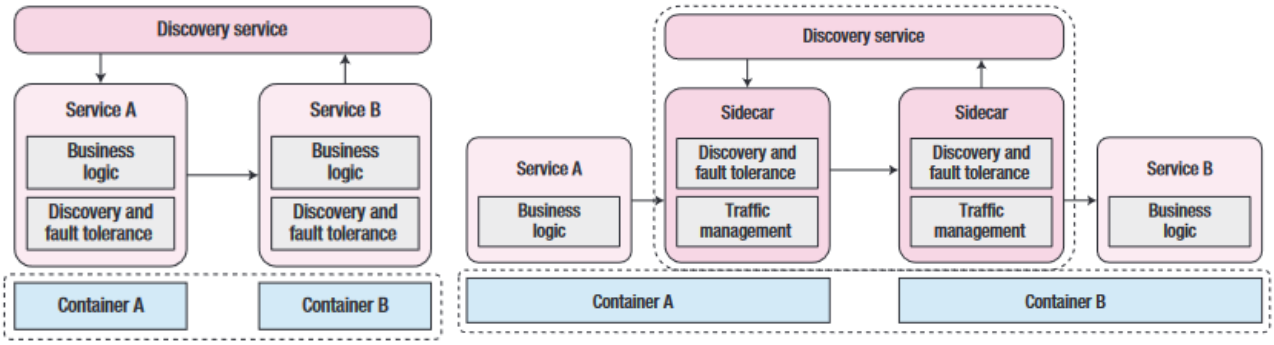


Figure 20: Two generations of microservice architectures (Jamshidi, Pahl, Mendonça, Lewis, & Tilkov, 2018)

On the left-hand side of Figure 20, the instrumentation is in the application (business logic) itself, or embedded in a library for communication, e.g., through remote procedure calls, used by the application. In the architecture on the right hand side, the instrumentation is in a sidecar proxy of which the application is completely unaware. Such proxies can also separate other important system services like service discovery, fault tolerance, and security. Creating and deploying sidecar proxies, e.g., using Envoy (Envoy, 2022), for a large number of microservices in a distributed architecture can be challenging and time consuming, which is addressed by service mesh technology (Li, Lemieux, Gao, Zhao, & Han, 2019), like Istio (Istio, 2022) or Linkerd (Linkerd, 2022). As shown in Figure 21, a service mesh is an infrastructure layer that sits on top of a microservice application and provides a uniform and efficient way to secure, connect, and monitor services by streamlining the configuration. Services communicate through proxies, forming a data plane, that is configured via a control plane to provide secure communication with encryption, load balancing, traffic policies, and observability (Li, Lemieux, Gao, Zhao, & Han, 2019) (Saleh Sedghpour, Klein, & Tordsson, 2022). This allows the effort of collecting telemetry data to be reduced and more effort to be spent on leveraging the data for monitoring or data analytics to add value to the system (Li, Lemieux, Gao, Zhao, & Han, 2019).

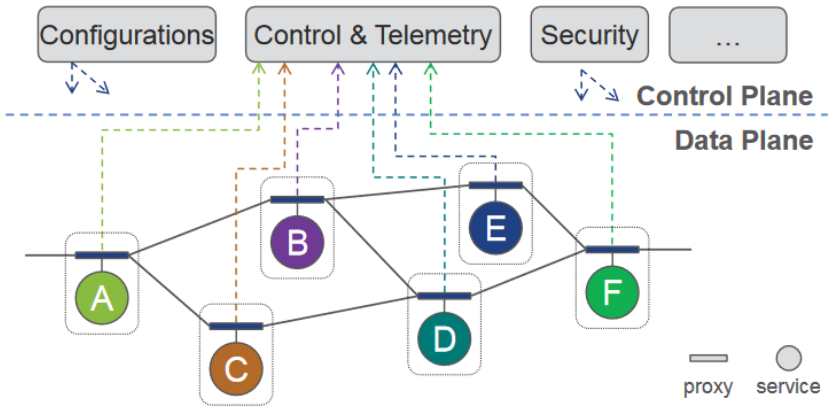


Figure 21: A service mesh logically comprises a data plane with microservices communicating through proxies and a configuration plane that manages and configures the proxies (Li, Lemieux, Gao, Zhao, & Han, 2019).

Solutions for Embedded Devices

There are also several mature tools and technologies from the embedded domain. Existing solutions are platform-dependent, making it difficult to find monitoring tools that can be used together in common

distributed environments. Unlike the tools from the cloud domain, the tools in this domain are focused on the verification process and not to the observability of the device in the deployment stage. The monitoring tools discussed above could potentially be used in the context of embedded devices following a black-box approach, as defined by Bhargavan et al. (Bhargavan, 2001). They propose to observe the inputs and outputs of the device from an external monitor and compare them against its specification. This approach can be useful in the case of a third-party device, whose software cannot be instrumented.

Rather, safety-critical systems should not only be observed from the point of view of their communications response, but also to internally verify their temporal and functional correctness. To this end, several companies are involved in the development of embedded monitoring tools with the aim of verifying the operation of safety-critical devices. In particular, among the proprietary license software, Percepio Tracealyzer (Percepio, 2022) is a trace tool suite that includes both a viewer and an embedded trace collector. It provides a detailed timeline of task scheduling and interrupts, system calls, as well as custom user events logged in embedded applications. Tracealyzer streams the trace data continuously to the host, either using a debugger JTAG or over device I/O interfaces like Ethernet or USB. Additionally, Rapita Systems provides RapiTask (Rapita Systems, 2022), a monitoring tool that collects information during program execution. It also represents data as a timeline of the tasks running in the embedded application, that allows verifying the scheduling behaviour of the system.

Regarding the open-source tools, one of the best known is LTTng (LTTng, 2022). It is a modular monitoring tool that enables integrated analysis of both kernel and user space on Linux-based systems. On the other hand, in the real-time operating system field, the RTEMS Trace Framework (RTEMS, 2022) is a native RTEMS module with the ability to instrument an RTEMS binary application and log its scheduler events at runtime, without rebuilding the code and without annotating the source code with tracepoints.

The embedded monitoring tools, described above, are intended to verify the temporal requirements of embedded devices as part of the V&V process. They are not designed to process traces and analyse metrics once the device is in the deployment phase, unlike microservice-oriented tools. For this reason, the methods and tools to be used at the device tier must be re-evaluated.

Conclusion

Clearly, there are many state-of-the-art open-source tools for providing observability and monitoring in a cloud environment. However, it is not easy to know which tools to select for a particular cloud platform and set of applications. The challenge is further exacerbated in the context of the compute (device-edge-cloud) continuum, since observability is required across all three tiers, and not just the public and private cloud for which the above mentioned were created. However, it is not just a matter of selecting appropriate tools for the compute continuum, but also considering how to configure, deploy and apply them. Separating instrumentation from business logic using a sidecar proxy creates a nice separation of concerns, but treating the application as a black box means that no information is collected about its execution, potentially impacting observability negatively. Deploying proxies furthermore gets complicated as the number of services increases. This can be resolved by streamlining configuration and deployment using a service mesh and create a higher level of abstraction, although this may result in more restrictions on the configuration of the system. Adding additional infrastructure layers is furthermore likely to negatively impact the performance of the system. Since the relevant technology has been developed quite recently, these trade-offs are not yet well understood for cloud applications, and even less for the complete compute continuum. Finding a solution that provides end-to-end observability and monitoring in the compute continuum remains an open issue.

4.2.1.1.3 Innovation step

Observability and monitoring in the compute continuum

Our first innovation step is to address the challenge of providing end-to-end observability and monitoring in the compute continuum. In terms of the TRANSACT reference architecture, this corresponds to answering the question of how to connect and coordinate the monitoring services across the tiers.

We propose to use a homogeneous technology stack across all tiers. This means using the same, or inter-operable state-of-the-art open source tools in all tiers to ensure that collected data is correlated and can be used together. Concretely, we will evaluate the use of OpenTelemetry, Prometheus, and Jaeger across the tiers to determine the boundaries of their applicability for different deployment options. Although best practices for observability (Goniwada, 2022) includes storing all telemetry data in a single location, this is not expected to be possible, as large amounts of data, some of which may be sensitive, would have to be exchanged between the tiers. Instead, we will investigate possibilities for distributed storage of collected data that can be securely exchanged between the tiers if necessary for analysis.

Exploring observability, performance, and development time trade-offs

As previously explained, there are different ways to configure and deploy observability tools in an architecture, e.g., by providing instrumentation in libraries, proxies, or as a part of a service mesh. Each option provides a different trade-off between application intrusiveness, performance, development effort, and the quality of the provided observability, resulting in many implementation options and no guidelines. We will address this problem by investigating the available options in each tier and experimentally quantify the trade-offs. Based on these results, guidelines for technology selection, configuration, and deployment will be proposed.

Automatic generation of monitors

In recent years, industry has focused on the device integration in distributed systems, which is not an easy task due to its wide heterogeneity. Model-based embedded code generation is a widespread practice that aims to improve productivity in device development. The models used can be both functional and non-functional, which generate a complete functional module or a code skeleton, respectively. The generated code is designed to be easily integrated with the edge and cloud services, including the mechanisms and protocols to communicate. In this case, we propose to use the non-functional models of the device platform and application to generate the skeleton of the code. This generated code implements the functionality to reproduce the modelled temporal behaviour, including activity precedence, thread allocation, priority assignment and other operating system settings.

The innovation step is based on instrumenting devices, incorporating the generation of monitors in the code generation process. Observability services in edge and cloud have the possibility of collecting the temporal behaviour from these devices using the common mechanisms of the rest of the system. This usually involves adapting monitoring data to be managed by the observability tools. This process often depends on the observability technology selected and the device communication protocol, for example, in case of Prometheus as monitoring service, if edge is connected to device using an MQTT messaging broker, there are several exporters to translate between MQTT topic representation and Prometheus metrics.

4.2.1.1.4 Application to use case(s)

The evaluation of the innovative steps will be done in an experimental platform that can be seen as an observability technology demonstrator related to observability and monitoring. The application workload will be a set of applications with different characteristics from DeathStarBench (Gan, et al., 2019) (Gan, et al., 2019), an open source benchmark for microservices.

The considered solution for observability is very general and should apply to most use cases in TRANSACT, with the possible exception for use cases with devices that are very resource constrained and limited to very light-weight instrumentation and monitoring (the exact boundaries for applicability will be determined as a part of our first innovation step). We will discuss the applicability of the proposed observability and monitoring solution in the context of UC4. In the 3D reconstruction scenario of this use-case, the end-to-end response time from when images are sent from the x-ray machine to the cloud for 3D reconstruction, until the response comes back into the operating theatre and is displayed on a screen must be within a given number of seconds with a probability given by the service level agreement (SLA). The cloud part of this requirement can be monitored straight-forwardly using the selected tools and technology. As a part of this research, we will investigate its applicability also in the edge and on the device. Concretely, we seek to transfer methods and insights from our observability technology demonstrator into the WP5 demonstrator, at least for UC4, in cooperation with the use case owners.

Additionally, the innovative step proposed in the automatic generation of monitors will be evaluated on the horizontal demonstrator. The objective of this solution will be not only to generate the necessary monitors to observe the temporal behaviour, but also to integrate the data obtained in the different services in the compute-continuum.

4.2.2 Performance modelling and prediction

4.2.2.1 AI-based performance modelling and prediction

4.2.2.1.1 Overview

As stated in Section 6.4.3 of Deliverable D3.1, the main purpose of an AI-based predictive model is to predict the future performance of the system. This prediction is paramount for management decisions that the performance manager makes. This requirement leads to challenges regarding safety-critical cyber-physical systems or cyber-physical systems in general. A model must *reliably*, *accurately*, and *timely* predict the performance metrics that the performance manager requires. The performance manager can then use these predictions for proactive performance management of the system.

The main challenge of integrating cloud computing services in safety-critical systems is the concept of “one-size fits all” underlying cloud computing solutions. This provides cost-effectiveness, but the reliance on the intricate structure of the internet for delivery and the universal sharing of resources make cloud computing a best-effort service. As a result, cloud computing can suffer from unexpected downtimes, which is inconsistent with some of the cyber-physical systems' requirements. Stable response time, reliability, or persistent availability are the requirements that can be jeopardized by these downtimes, even though it may be possible to achieve statistical guarantees over extended periods. The root cause of service-latency fluctuations is often complex, and we are given very little information about the inner workings of cloud and internet infrastructure. It is possible to use white box modelling to analyse known factors in cloud service delivery. However, it can be too challenging to use them to proactively analyse and manage fluctuations in service latency caused by factors we have little control and information over. Thus, black-box, data-driven models capable of learning a predictive service-latency model can provide added value. This way of modelling should enable accounting for the complexity and lack of information about the cloud provider infrastructure by automated learning from observations.

One goal of performance modelling is to obtain accurate predictions for (future) end-to-end latency of cloud services. But given the nature of both the cloud systems and of the black-box prediction models, it will not always be possible to assess the value or accuracy of the obtained predictions. Hence, a secondary goal is to quantify the uncertainty in latency predictions. Uncertainty may arise from inaccuracy in learned models, lack of information during inference, or run-time data that deviates substantially from training data. The

performance manager can make better decisions by quantifying the uncertainty in latency predictions. For instance, it can make more conservative decisions in presence of high prediction uncertainty.

4.2.2.1.2 State of the art

One of the main challenges in deploying safety-critical systems in distributed platforms is ensuring that the application can meet the deadline while the cost is minimized. A general approach to this problem is to use scheduling methods along with resource management techniques. Scheduling and resource management in a system with dynamic resources requires to have an estimation of system performance and behaviour.

Several methods concerning performance prediction in cloud-based platforms have been proposed. These can be classified into white-box and black-box categories. In the white-box methods, the performance of the system is estimated through analytical approaches like queueing theory, (Ardagna, et al., 2021), (Urgaonkar, Shenoy, Chandra, Goyal, & Wood, 2008) and regression (Bulej, et al., 2021) . However, the complexity of dynamic resource availability and the dependencies between components of a distributed system can be challenging for white-box methods. This makes it difficult to provide accurate online performance predictions for these system with typical analytical methods alone.

The black-box approaches encompass AI (Artificial Intelligence) models, such as (deep learning) neural networks. These approaches relate observed resource metrics and history of application performance with the predicted performance of future application calls on a specific metric. AI models have proved to be very efficient to comprehend complex systems. These models can in some cases reach a remarkably good performance in terms of prediction accuracy. Ideally, these models should be able to effectively predict future service-latency using the given data. Unsurprisingly, despite the interesting advantages of AI, AI methods also have downsides. As for white-box approaches, the most challenging aspect of AI-based models is to reliably and accurately model the performance in the safety-critical application under dynamic and evolving circumstances. It becomes even more challenging when we realize that the circumstances, we are trying to model involve the intricate structure of the internet.

Most classic methods of predicting performance involving the internet structure only used the black-box approach. For time series prediction, methods like state-space models (hidden Markov models) (Fuh & Tartakovsky, 2018), the AutoRegressive Integrated Moving Average (ARIMA) model (Calheiros, Masoumi, Ranjan, & Buyya, 2014), Support Vector Regression (SVR) (Zhang, Zhou, Chang, Yang, & Li, 2013), and linear Back-Propagation Neural Networks (BPNN) (Kumar & Singh, 2018) were used to predict latency for cloud services. (Rahman & Lama, 2019) applied various machine learning techniques including support vector machine, linear regression, and multilayer perceptron to predict the end-to-end tail latency in the presence of performance interference. (Yan, Liang, Lu, Wu, & Zhang, 2021) proposed a method to predict latency over time using an attention-based Bi-LSTM (Bidirectional Long-Short Term Memory neural network). An AI model composed of convolutional neural networks and boosted trees was proposed in (Zhang et al., 2021) to predict service response time both for short-term and longer-term prediction horizons; the boosted decision trees are used to predict whether given latency bounds will be violated. The goal of the work was to manage the resources and control the quality of service for cloud services. Runtime performance prediction for high-performance computing systems with state-space models targeting accurate and fast online prediction was proposed in (Naghshnejad & Singhal, 2020). (Bankole & Ajila, 2013) presented a machine learning-based technique to manage the resource provisioning in the cloud with neural networks, support vector machines, and linear regression. Yet, none of the work presents a proper solution to the unreliability of models whereas they are applied to dynamic systems with unforeseen application deployments.

Besides pure white-box and pure black-box approaches, there are also hybrid grey-box models that combine first-principal modelling and data-driven modelling. (Nguyen, Alesawi, Li, Che, & Jiang, 2020) proposed a grey-box fork-join queuing model with black-box fork component for modelling and prediction of cloud

computing service performance, including latency. (Crankshaw, et al., 2020) use discrete-event simulation in addition to an adaptive monitoring to predict and manage cloud service performance.

AI-based performance prediction relies on trained models based on historical data in a specific test environment. This makes the resulting models especially susceptible to overfitting and provides poor generalization capabilities for out-of-domain data and different environments. In safety-critical systems, besides having accurate predictions, it is important that a model is not overconfident and thus misleading. Thus, as stated by (Amini, Schwarting, Soleimany, & Rus, 2020), it is important to quantify uncertainty in model predictions when it comes to safety-critical systems.

To the best of our knowledge, the problem of prediction of performance metrics with uncertainty quantification using deep learning has received no attention in the cloud-computing domain so far. How a machine learning algorithm extracts models from data causes uncertainty in its various facets. As a result, it is highly desirable to have a reliable representation of uncertainty where having a trustworthy model is essential. Uncertainty quantification (UQ) (Parry, 1996) (Der Kiureghian & Ditlevsen, 2009) is a probabilistic method to model uncertainty in a way that complements uncertain models such as machine learning and deep learning ones (Hullermeier & Waegeman, 2021). The result of UQ is then a probabilistic prediction interval for the estimated metric. Uncertainty is usually classified by its sources and is divided into two categories:

1. *Aleatoric uncertainty*, or uncertainty resulting from (randomness of) the data
2. *Epistemic uncertainty*, or uncertainty resulting from (ignorance of) the model

Aleatoric uncertainty is also called the irreducible part of the uncertainty because we do not expect more data to reduce the uncertainty due to inherent randomness, as opposed to epistemic uncertainty. In most cases, these two sources are not distinguished in the final representation. However, it is important to distinguish them when developing a method to quantify uncertainty (Hullermeier & Waegeman, 2021).

Regression-based neural networks are being increasingly used in safety-critical applications such as robotics, control and real-time computer vision (Cal, Wang, Huang, Liu, & Liu, 2021). These applications have in common the ability to infer a model that can reflect on its own (in)accuracy. Quantifying uncertainty is necessary for the wide-scale adoption of AI methods in safety-critical applications. Uncertainty estimates can be used to assess the confidence of the model's predictions and capture test samples' domain shift.

Aleatoric uncertainty can be inferred directly from data characteristics such as noise. Epistemic uncertainty cannot be inferred from a model alone. As such, it is significantly more difficult and essential to quantify epistemic uncertainty. Quantification of epistemic uncertainty has been extensively studied and can be categorized into two different approaches (Abdar, et al., 2021):

1. In Bayesian-like approaches, a randomized model is sampled during inference. This, in the case of machine learning, results in creating virtually different neural networks from a single randomized one. In Bayesian deep learning (Wang & Yeung, 2016) uncertainty is quantified by sampling the random instances and observing the degree to which they agree. Bayesian regularization, like Monte Carlo Dropouts (Gal & Ghahramani, 2016), uses randomized regularization techniques during inference for sampling and quantification of uncertainty.
2. In deterministic approaches, the model is multiplied by various heuristics instead of having any randomness in a model. For example, in deep Ensembles (Lakshminarayanan, Pritzel, & Blundell, 2017), data is perturbed for each model copy using a heuristic. Adversarial examples (Goodfellow, Shlens, & Szegedy, 2014) is one of the methods that maximize difference between models' results by choosing perturbations that maximize models' confusion during inference. An example of its

usage can be seen in Figure 22. Subsequently, the results can be combined using simple statistics for calculating the mean and variance of the models' mixture of predictions to quantify uncertainty.

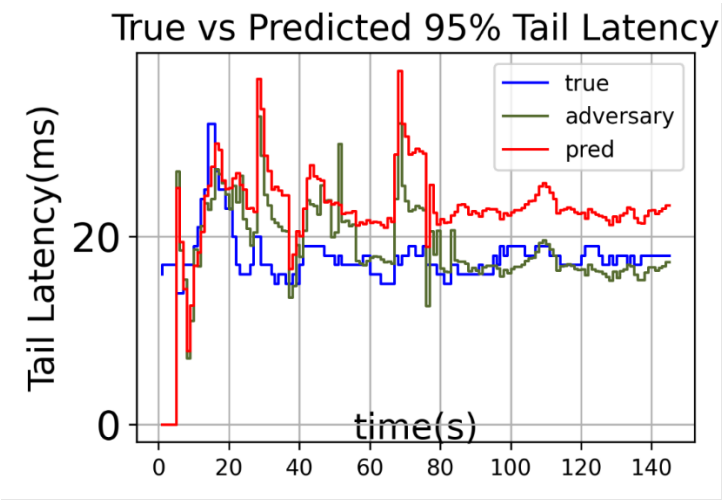


Figure 22: (Zhang, Hua, Zhou, Suh, & Delimitrou, 2021) Latency prediction framework showing predicted 95% tail-latency augmented with adversarial examples (Goodfellow, Shlens, & Szegedy, 2014).

Generally, the approaches mentioned above for epistemic uncertainty quantification become more challenging by increased demand for computing power as the number of models or repetitions used for uncertainty assessment increases.

There are recent works that rely on heuristic quantification of uncertainty by augmenting the analogous probabilistic neural network model. For example, deep evidential learning (Amini, Schwarting, Soleimany, & Rus, 2020) replaces the common gaussian distribution assumption with a gaussian distribution with unknown mean and variance. This results in representing uncertainty by a single model, eliminating the need for model multiplication. Consequently, these methods will not need additional computational power to quantify uncertainty. However, by pushing the need of having multiple models for diversity into the distribution itself, these approaches could run into the risk of lower-level underfitting and instability, resulting in uncertainty estimates that are too wide to be taken seriously.

4.2.2.1.3 Innovation step

AI methods have been demonstrated to have the ability to capture the complexity and dependencies in distributed systems such as cloud services, which makes it a suitable means to model and predict the performance in the cloud-based platforms. Despite the advantages the AI has brought to this area of research, there are still gaps in performance modelling and prediction using AI methods. Data dependency of the AI methods, in terms of quantity and quality, is a source of these gaps and challenges, as it needs the right data to be efficiently employed.

The first downside of the data dependency of AI models is that data should contain enough samples to train the models to reach sufficient accuracy. This can be quite a challenge due to lack of data in the beginning of a production. Another downside is that AI models are trained based on the data given in the training phase. This data only reflects the system information at the time of training deployment. If, during runtime, the deployment of the application changes to new sets of resource configurations that have not been seen in the training phase, a domain shift, then the model may lose its ability to predict the performance accurately.

A source of performance inefficiency in AI-based models are application redeployments at runtime. A goal of our proposed method is to maintain the model performance during runtime. To this end, the model should

have a mechanism to be able to adapt to new deployments and get updated as soon as possible to prevent the performance inefficiencies. A potential solution to this problem is online transfer learning in which the model is updated at runtime. There are some issues with these methods such as variation in model performance and runtime training overhead that need to be addressed.

Another possible solution may be found by combining the AI-based modelling (known as a black-box approach) with analytical models (known as white-box approach) and present a combined ('grey-box') approach to model the system. AI models can understand a system through features that are fed to the model. We intend to use analytical approaches to analyse the system resource behaviour and its effect on the response-time, obtain an abstraction of these pieces of information and feed them as improved features to the AI-model. As a result of this method, whenever there is a change in the system resources, effects of this changes are first analysed for features which are then given to the model to predict the performance.

Since our applications are real-time (safety critical) systems, the end-to-end response-time of the application is considered as a metric representing the performance of the system. The end-to-end response-time stems from various sources of delay including network response time, queueing, and execution time. The causes of these delays can be different, so each delay is analysed separately. With a grey-box approach, one analyses the resources and background process and their effect on the execution time, considering dynamic resources and application redeployments. Then, we can extract the relevant features and use them in a predictive model.

Besides prediction of the primary metrics, we intend to research if it is possible to effectively assess the uncertainty in the predictions to improve dependability, especially during initial stages. Towards this goal, we first aim to quantify the uncertainty in current machine learning and deep-learning benchmark models that are popular in the cloud computing domain.

Our next step is to complement existing works with different methods of uncertainty quantification and benchmark them. In the end, we want to achieve a trustworthy model essential to functioning of cyber-physical systems without unacceptable levels of performance sacrifice.

4.2.2.1.4 Application to use cases

The application of our proposed approach has been evaluated in relation to TRANSACT UC4. To start with the evaluation of our method, a modified and simplified version of UC4 has been implemented to provide us with sufficient data as input to the AI learning and the analytical parts.

This application will be further developed towards a generic Kubernetes (Kubernetes, 2014) based microservice architecture, divided into three steps.

1. First step concerns implementing a simple prototype that can replicate the behaviour of UC4 to some extent. In this implementation, regular optical cameras are employed instead of medical image sensors and the medical 3D reconstruction application is replaced by a non-medical, open-source 3D reconstruction application, OpenSFM (Mapillary, 2021). These modifications let us monitor the system metrics and traces using existing monitoring tools such as Prometheus (Prometheus, 2012) and OpenTelemetry (OpenTracing, 2022), generate required data, and freely make changes to the setup. Furthermore, for the future we consider the development of the application into an architecture consisting of four parts as seen in Figure 23. The broker is supposed to store workflows for broadcasting while the database is supposed to store data which is retrieved by its own dedicated service. A translator module is provided to enable communication with the broker.
    a. Gateway: The part that serves the user, such as a webserver. Template webservers are widely developed and available examples on the internet are numerous.

b. Algorithms and pre-registered routines as services, such as the OpenSFM 3D reconstruction. Existing algorithms and routines that need to be repeatedly executed and scaled will be wrapped as containers. These containers will be accompanied with a complementing translation module to enable them to communicate with other containers. They will then be deployed as services to be handled by Kubernetes cluster management.

c. A backend: This part is responsible for translating user requests into meaningful workflows that need to be done by other services. This is the part that should be actively developed and custom tailored according to the requirements.

d. Databases and Message brokers. Since (micro)services are stateless on their own, the backend should be assisted by corresponding databases and message brokers. These extra applications will keep the workflow active by storing the entire workflows and broadcasting them to the responsible services every turn.

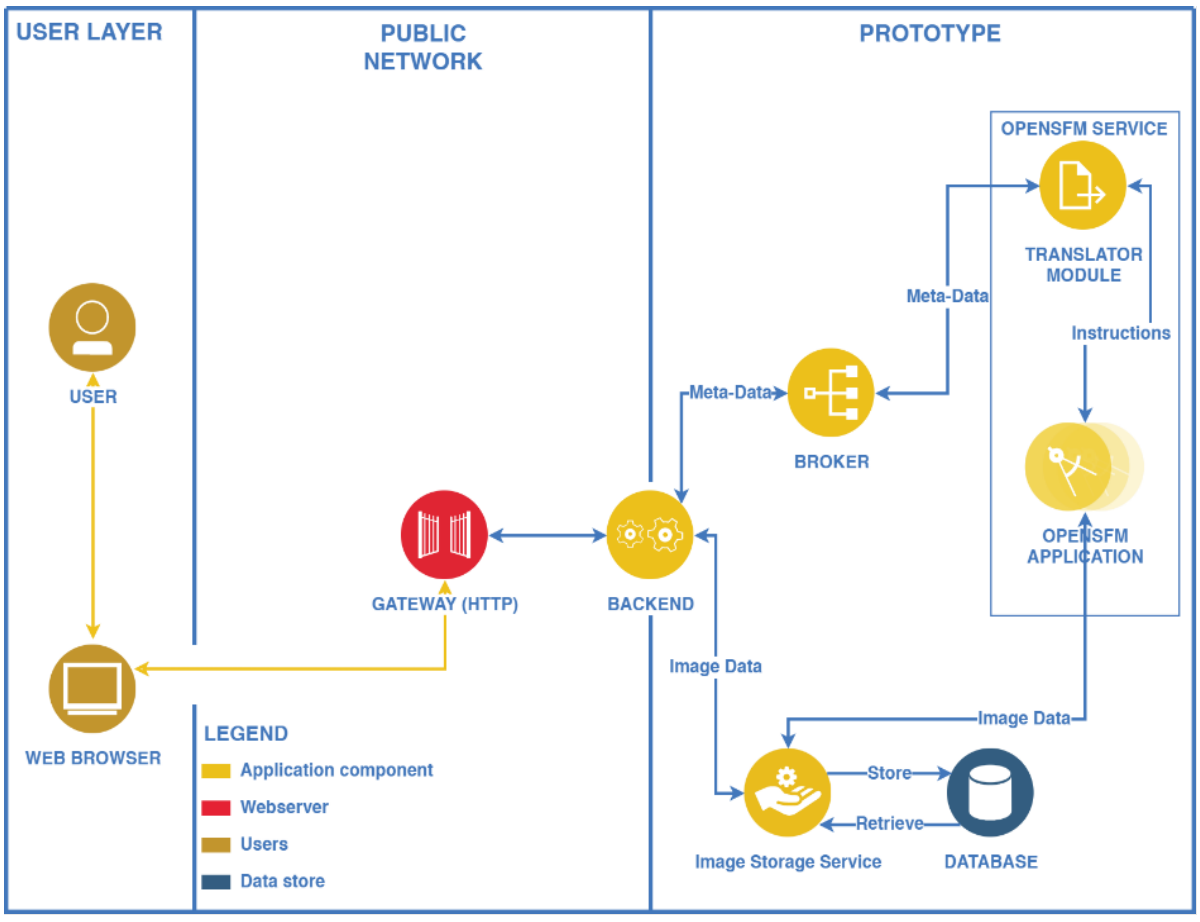


Figure 23: The architectural overview of the example prototype.

The placement of each one of these parts and their components can be decided by configuration of the corresponding Kubernetes cluster. For the algorithms part, different algorithm implementations such as 3D reconstruction (OpenSFM) or deep learning inference can be used.

2. The second step concerns creating a simple foundation to compare different methods of prediction and of quantifying the uncertainty of predictions.

3. Third step concerns synthesizing a reliable dataset that can be used to train and compare the different neural network architectures. For this section, we can initially rely on collecting historical performance data from other published microservice applications such train-ticket (Zhou, et al., 2021) and DeathStar (Gan, et al., 2019). After the completion of the prototype of UC4, we can gather more data from it to further enrich our dataset.

For prediction of performance metrics, it is important to first recognize the potential parameters of the system affecting the execution time of the application. Since (possibly virtualized) hardware resources i.e., CPU and memory, are the most obvious parameters influencing the time of a process we start with gathering information of those resources and of the resulting execution times.

First, a trace of execution times has been inspected and a time dependency in the samples has been observed indicating a time-series analysis of the data is beneficial in this case. We then pre-processed the monitoring information from resource utilization to create new features such as average available resources and background workload for each specific process. In the next step, a feature selection process is applied on the generated features to determine the importance of the features in execution time prediction. It is also observed that the selected features have a high impact on the execution time.

In addition to choosing the right features in AI models, choosing an appropriate model or algorithm impacts the prediction performance. An AI model should be able to comprehend the complexities and dependencies in the distributed systems and capture the existing time-dependency in the data. LSTM is an obvious choice here because of its ability to detect the long-term time dependencies. However, due to lack of sufficient data, LSTM is not our only solution here.

Next to the LSTM model, decision trees and deep-learning methods (convolutional neural networks-CNN) have also been evaluated using the extracted data from our setup. The graph in Figure 24 shows observations and predictions of response times from a LSTM, CNN and decision tree model for training and for test data sets. The Y axis shows the response-times in seconds and the X axis shows the sequence of requests. It is observed that the predictions follow the pattern of actual data with a high accuracy. The mean absolute error is 0.127 and 0.162, 0.631 and 0.66, and 0.005 and 0.186 for train and test data in LSTM, CNN, and Decision tree, respectively.

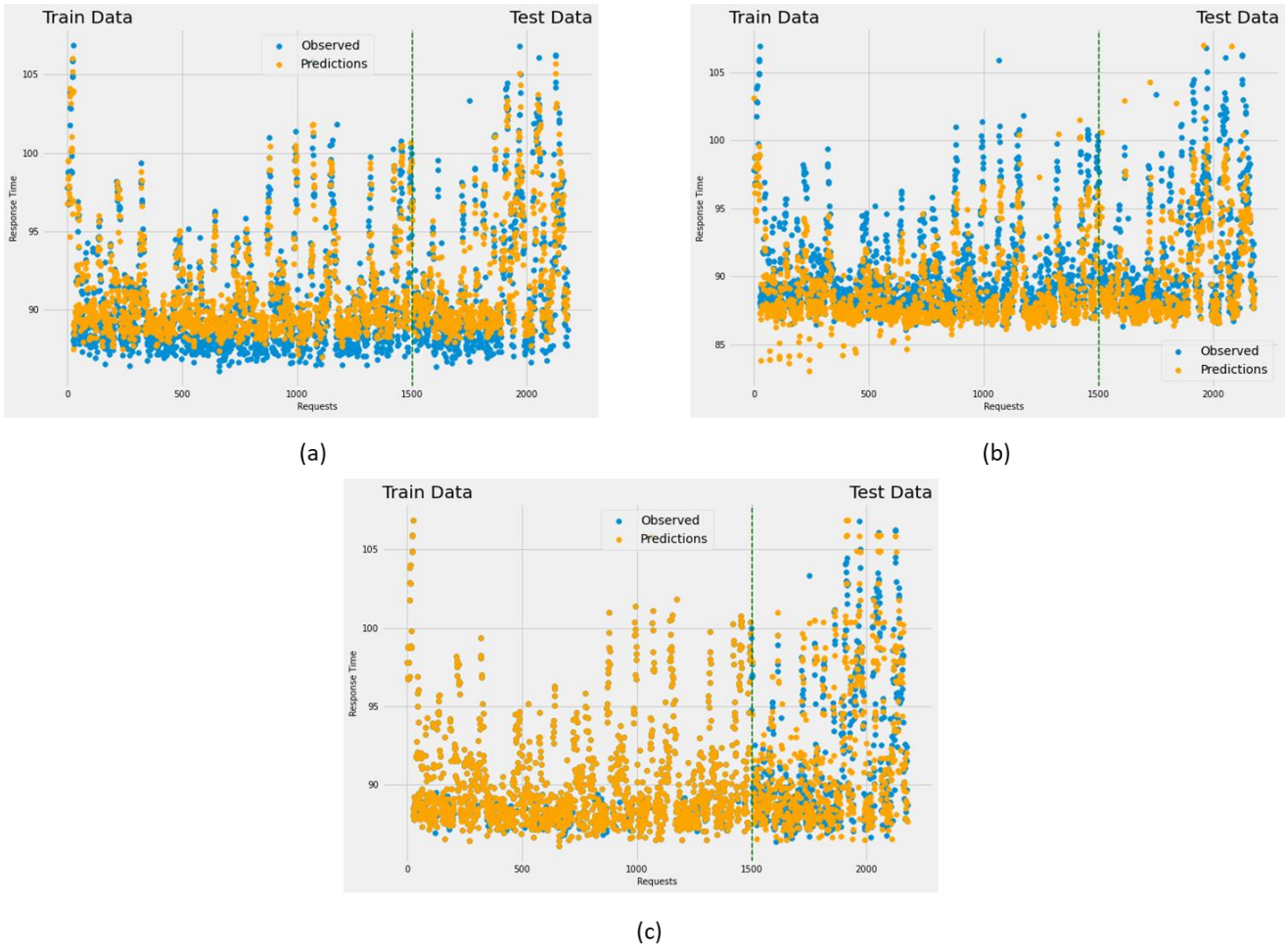


Figure 24: observation and prediction data in train and test phase for (a) LSTM, (b) CNN, (c) Decision tree

For uncertainty prediction, we intend to reimplement existing methods such such as (Zhang, Hua, Zhou, Suh, & Delimitrou, 2021) and (Qiu, Banerjee, Jha, Kalbarczyk, & Iyer, 2020) (Zhang, Hua, Zhou, Suh, & Delimitrou, 2021) and using a deep learning framework like Tensorflow (Tensorflow, 2015) in the first step. This framework enables a faster and easier development and prototyping since it is abstract and well documented.

In the next step, we intend to augment the neural networks in these works to enable them to quantify uncertainty. For this purpose, we specifically aim to pick one candidate method of uncertainty quantification for each approach, namely Monte-Carlo Dropouts (Gal & Ghahramani, 2016), Deep Ensembles (Lakshminarayanan, Pritzel, & Blundell, 2017), and Deep Evidential regression (Amini, Schwarting, Soleimany, & Rus, 2020). For comparison, we introduce the minimum number of predictions captured within an interval in a fixed time frame as a new metric named tolerance probability. Since point estimations of previous works do not have any intervals, we set fixed intervals as the base point of comparison. In the final step, we compare the tolerance probability and the range of fixed intervals vs. the prediction intervals introduced by various uncertainty quantification methods. We argue that this comparison will present a more stable and trustworthy model among the candidates.

4.2.2.2 Simulation-based performance analysis

4.2.2.2.1 Overview

In this section, we present the current status of a methodology for simulation-based performance analysis and management of distributed safety-critical cyber-physical systems. This refines the concepts for "Performance modelling and prediction" from deliverable D3.1. As mentioned in that deliverable, modelling and performance prediction plays a role at design time and at run time of distributed safety-critical cyber-physical systems. The proposed methodology is centered around discrete-event simulation models for system performance. The main method evolves and uses a system performance model during the system's lifecycle. I.e., the model comes into being during the architecture phase, evolves further during development, and ends up as a runtime component for performance management. In this deliverable, we focus on presentation of the methodology, and on the first step of the main method. In the second version of this deliverable D3.5 that is due in M33, we refine this work.

4.2.2.2.2 State of the art

Microservices are an architectural style that is similar to service-oriented architecture. It emphasizes light-weight technologies and technical boundaries (each microservice is implemented and operated by its own team). Microservice-based systems are typically deployed in the cloud. Even though system performance often is important, overviews such as (Dragoni, et al., 2017) and (Jamshidi, Pahl, Mendonça, Lewis, & Tilkov, 2018) on microservices only marginally recognize performance, and papers on microservice-based CPS such as (Thramboulidis, Vachtsevanou, & Solanos, 2018), (Buchgeher, Ramler, Stummer, & Kaufmann, 2021), (Gartziandia, et al., 2021) also give little or no attention to performance. The work in (Heinrich R. , et al., 2017), however, gives an overview of performance-engineering challenges for microservices in the areas of performance testing, monitoring, and performance modelling. A recent overview paper (Zeng, et al., 2022) discusses performance optimization in the microservice era and distinguishes three areas: performance evaluation and monitoring, resource provisioning and management, and system tuning and optimization. Modelling and prediction is regarded in the context of resource provisioning and management. The work of (Gan & et al., 2019) discusses, amongst others, the impact on performance of moving from a monolithic system to a microservice-based system. The overviews (Heinrich R. , et al., 2017) and (Zeng, et al., 2022) agree that finding proper models and abstractions for performance engineering of microservice-based architectures is a research challenge.

There is quite some work on performance modelling and prediction for cloud-based systems (not restricted to microservices or CPS) which at least partially addresses the challenges of (Heinrich R. , et al., 2017) and (Zeng, et al., 2022). The survey in (Amiri & Mohammad-Khanli, 2017) suggests that only a few *simulation-based* methods for cloud systems exist. The following have been found. In (Gribaudo, Iacono, & Manini, 2017), an ad-hoc simulator is used to analyse abstract micro-service applications. In (Cuomo, Rak, & Villano, 2015), discrete-event simulations of cloud-based systems are used to predict application performance. In (Idziorek, 2010), an ad-hoc simulation model is used to study horizontal scaling strategies. In (Calheiros, Ranjan, Beloglazov, Rose, & Buyya, 2011), the CloudSim toolkit is introduced, which enables modelling and simulation of Cloud computing systems and application provisioning environments. A dedicated simulator for micro-service based systems is presented in (Zhang, Gan, & Delimitrou, 2019). In (Turin, et al., 2020), a model of the Kubernetes environment is presented in the ABS modelling language. In (Etemad, Aazam, & St-Hilaire, 2017) and (Al-Zoubi & Wainer, 2020) DEVS models of cloud systems are created. In all these works, except for the last two (using ABS and DEVS), the formal semantic foundation is unclear. Furthermore, none of the mentioned work supports stochastic processes. There is, however, a DEVS extension that supports Markov modelling (Seo, Zeigler, & Kim, 2018).

Calibration and validation of system performance models is a topic that gets little or no attention in the research mentioned above, even though it is an important and non-trivial part of the modelling process. Some publications in the area of performance modelling are (Pimentel, Thompson, Polstra, & Erbas, 2008) and (Parappurath, Voeten, & Kotterink, 2013). In other domains, however, model calibration is well studied. See, e.g., the Bayesian framework proposed in (Kennedy & O'Hagan, 2001). This is carried to an online setting of continuous calibration in, e.g., (Ward, Choudhary, Gregory, Jans-Singh, & Girolami, 2021) and (Chong & Song, 2019).

4.2.2.2.3 Innovation step and solution description

From the related work, we conclude that many performance-engineering ingredients for cloud-based systems are well-studied, and that work specific for the cyber-physical-system domain starts to emerge. Work that is based on discrete-event simulation techniques, is sparse, however. An advantage of simulation is that it often does not immediately depend on measurements or detailed operational data. Typically, high-level workload and system models can be created based on initial assumptions or back-of-the-envelope calculations. These can be adjusted (calibrated and validated) when data becomes available further on in the development process. Another advantage is that a simulation run (a single behaviour of the modelled system) can be manually inspected to gain insight in the details of dynamic behaviour. We therefore investigate simulation-based performance analysis and management for distributed safety-critical cyber-physical systems, and systematize it as a methodology in the sense of (Theelen, 2004) and (Heemels & Muller, 2006). The methodology consists of (i) formalisms, (ii) techniques, (iii) main method and sub methods, and (iv) tools to solve the problem at hand. The innovation steps are that we adopt existing solutions to the TRANSACT context in the form of an integrated methodology (ongoing), and that we mature solutions where needed (ongoing). Below, we present the current state of the methodology, and start with the main method, which consists of application of performance modelling and analysis during three phases of the system's lifecycle:

Performance models for architecture and high-level design

Abstract performance models are constructed that are "engineering models" in the sense of (Lee E. , 2017). That is to say that the intent of our future system is expressed in the model: the model serves as a specification (at least w.r.t. performance) of what is yet to be built. Early *exploration* using the model can take place to investigate alternative architectures and designs. Such early explorative work can be valuable, even though the model has not been validated against reality (that is not there yet). It can show qualitative trends, potential performance risks, etcetera. It increases the understanding and insight of the user in the dynamic behaviour. This is far from trivial as performance often is a cross-cutting, and therefore emergent property (Heemels & Muller, 2006), (Derler, Lee, & Vincentelli, 2012), (Fitzgerald, Larsen, & Verhoef, 2014).

Performance models for detailed design and implementation

The performance models from the previous phase are used as guidelines (in addition to other design artifacts, of course) to realize the system components. Once one or more of these components are ready, *measurements* can be gathered about their runtime behaviour (e.g., nominal execution times). This allows us to calibrate and validate (parts of) the performance models. The role of the models is gradually shifting from "engineering model" to "scientific model" (Lee E. , 2017). The latter means that if there is a significant mismatch between the model and the data, the model is wrong and should be adjusted. Such a mismatch may reveal wrong assumptions, which may lead to corrections in the design. A corrected, partially validated, model can then be used again to further *improve* the design. The development process is not linear: due to increasing knowledge and insight we may have to revisit earlier choices.

Performance models for operation

We envision to use the calibrated and validated performance models online for model-predictive control to ensure safety and performance of the running system. Monitoring gives us a steady stream of data that

allows *continuous calibration and validation* of the online performance models. Various models may also run together. For instance, a model that predicts workload (inferred from historical data) can be used to generate a workload for a discrete-event simulation model that predicts the system performance for a certain bounded timespan in the future.

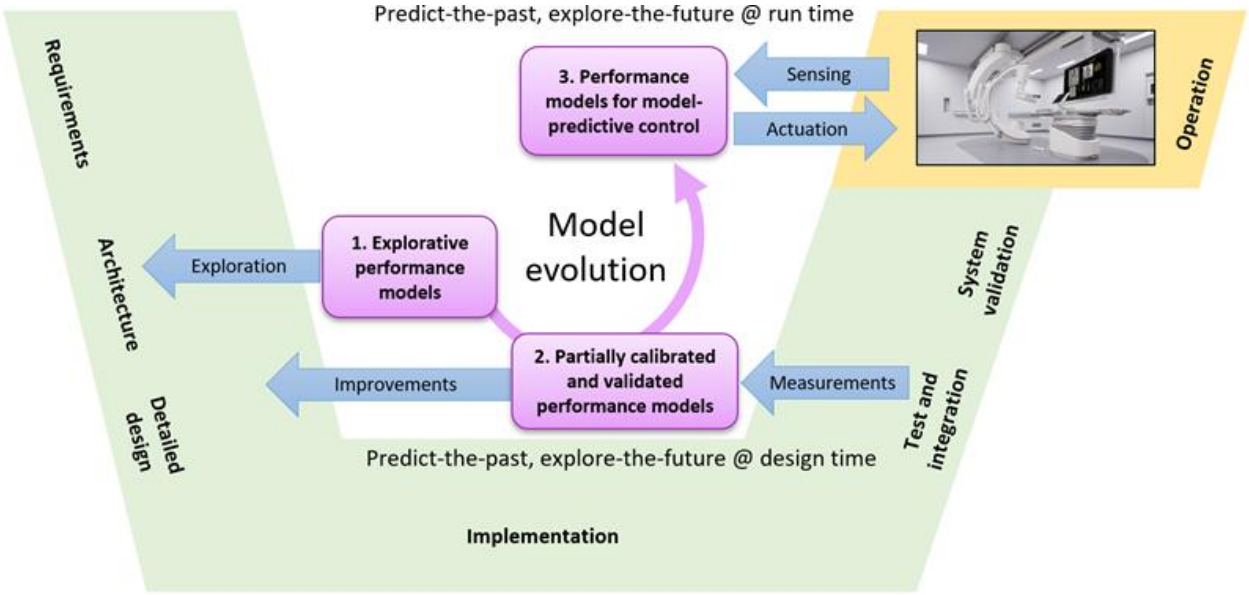


Figure 25: Main method: using performance models during the lifecycle.

Figure 25 visualizes the main method of the methodology. Note that the *predict-the-past-explore-the-future* paradigm (Parappurath, Voeten, & Kotterink, 2013) is central to the successful use of performance models, both at design time and at run time. Its essence is that data of (parts of) an existing system is used to calibrate and validate the model. The model (and variations of it), which have earned a certain amount of trust through the previous validation step, can then be used for exploration to predict future behaviour in various scenarios and modes of operation.

The main method described above is supported by sub methods that refine certain steps, and by formalisms, techniques and tools that are used to perform the tasks described in the methods. At this moment, we have identified and used the following:

Formalisms: We use the Parallel Object-Oriented Specification Language (POOSL) for modelling (Voeten, 1995). It has a formal semantics based on Markov chains. This allows flexible yet mathematically precise modelling (e.g., probabilistic choices for workload modelling), and formal analysis of performance metrics through discrete-event simulation (Theelen, 2004). Furthermore, we use formalized execution traces to visualize and analyse a single execution of a model (Hendriks & Basten, Performance Engineering with Trace, 2018).

Techniques: The main techniques are discrete-event simulation with formal performance analysis, and visualisation of single model executions.

Tools: We use the Eclipse open-source POOSL tooling (POOSL toolset, 2022), which supports modelling of parallel processes that communicate through message passing. It has formal semantics based on Markov chains, advanced constructs to model data, time and synchronization, and a high-performant discrete-event simulation engine. The Eclipse TRACE4CPS (Eclipse TRACE4CPS, 2022) tooling is used for visualization of single execution traces of the model.

Sub methods: We have applied various modelling patterns from other domains. These are the Y-chart pattern (Balarin, et al., 1997) (Kienhuis, Deprettere, Vissers, & Wolf, 1997), and a piecewise-linear progress abstraction for resource sharing (Hendriks, Basten, Verriet, Brassé, & Somers, 2016). Furthermore, we applied a workload abstraction that makes the model more scalable, which typically is needed for cloud-based systems with many users.

Summarizing, we adopt existing solutions to the TRANSACT context to work towards an integrated methodology for simulation-based performance analysis and management for microservice-based CPS. Furthermore, as part of the TRANSACT work, we have contributed to maturing the POOSL methodology and applied it to distributed safety-critical cyber-physical systems. The POOSL toolset is now open-sourced to the Eclipse Foundation and professionally supported by OBEO (POOSL toolset, 2022).

4.2.2.2.4 Application to use case(s)

The first step of the main method has been applied to TRANSACT UC4. That is, we have created a high-level performance model of UC4 with the POOSL toolset for the purpose of architectural exploration. The model follows the Y-chart pattern (Balarin, et al., 1997), (Kienhuis, Deprettere, Vissers, & Wolf, 1997)] and includes a piecewise-linear progress abstraction for shared bandwidth (Hendriks, Basten, Verriet, Brassé, & Somers, 2016). Furthermore, a stochastic workload is used, which results in a burst load for the communication and compute platform.

UC4 scales up to approximately 2000 hospitals, each with five operating rooms. Our first model cannot be analysed for these parameter values within reasonable time. We have created a second model which abstracts the workload of a subset of the hospitals. This model scales much better and the mentioned parameter values can be used.

The POOSL toolset supports model creation by a textual domain-specific language. It also has means to visualize the models. Figure 26 shows, for instance, the structure of the Cloud cluster class of the model. It consists of an S3Bucket, an Orchestrator, a ReconstructionDistributor, a number of Reconstructors and a WebApp. These are all POOSL process classes that define the behaviour. The cluster-class concept is a means to encapsulate all this. That is, the Cloud cluster class can be used to construct the full model (including the hospitals) without explicit references to its contents. This facilitates modular models.

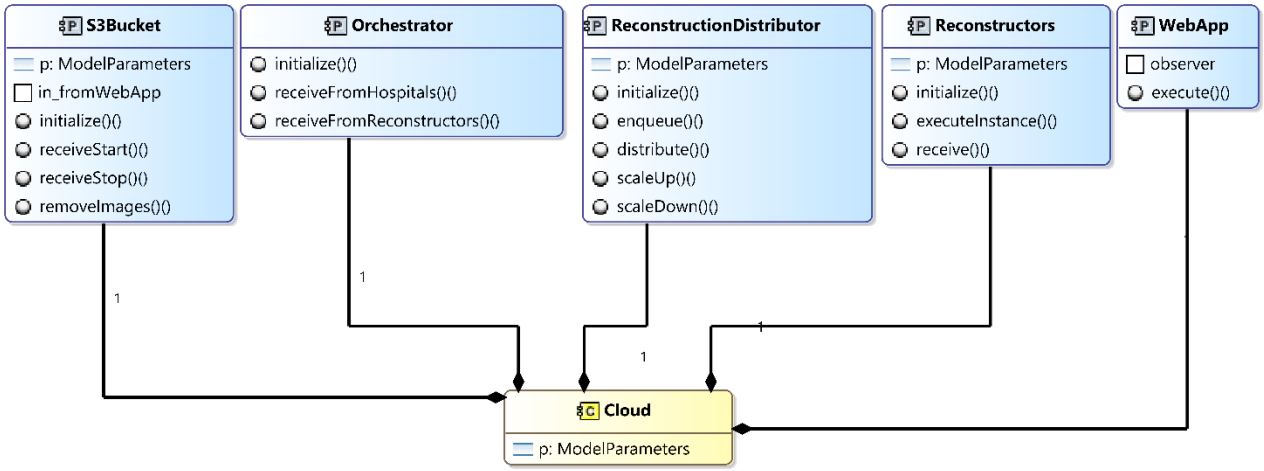


Figure 26: The structure of the Cloud cluster class.

The POOSL simulation can generate input for the TRACE4CPS tool, which visualizes the dynamic behaviour of the model (the model has to be manually instrumented to generate meaningful data for the TRACE4CPS tool).

Figure 27, for instance, shows the TRACE4CPS view of a simulation run. This clearly shows the burstiness of the load that is generated on the cloud infrastructure. Note that the model is heavily parameterized. All parameters are encapsulated in the ModelParameters data class. Again, such encapsulation facilitates ease of modification.

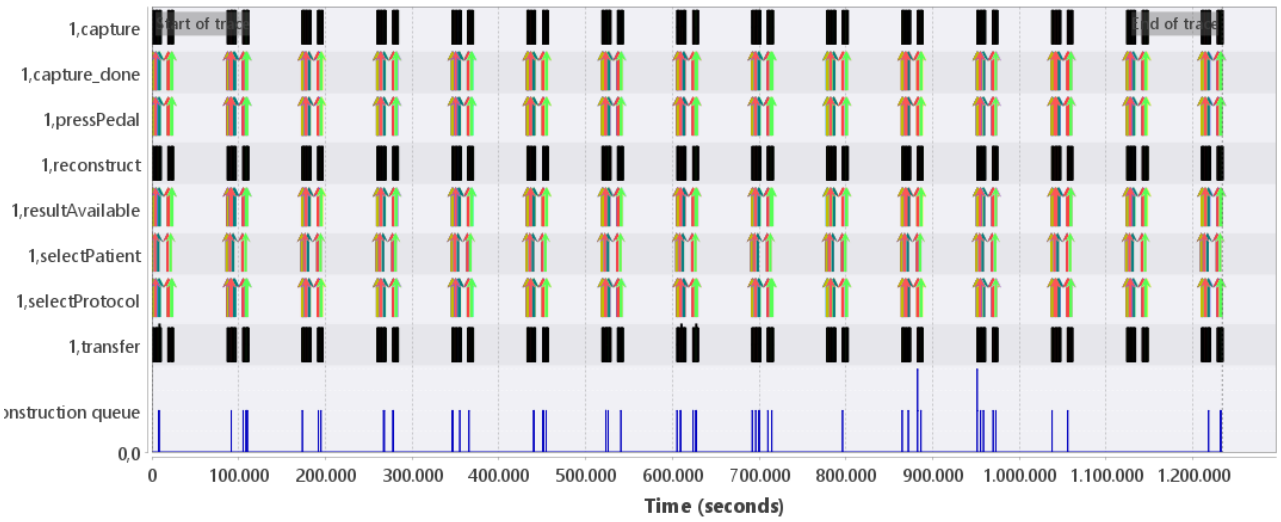


Figure 27: A simulation trace for 1 hospital with 5 rooms that each receive 5 patients per day during 15 days.

This modelling and analysis exercise shows us that:

- Natural modelling of distributed safety-critical cyber-physical systems on a high level of abstraction with POOSL seems feasible. Modelling patterns from other domains can be reused, and we see more cloud-specific modelling patterns emerge. For instance, explicit modelling of a single user and abstraction of the load on the cloud system by other users. Typical response-time properties can be formally expressed and analysed. Behaviour of the management infrastructure can be modelled (e.g., auto scaling). Visualization of single model executions by TRACE4CPS adds value. Figure 28, for instance, shows why a certain response time is high (approximately 48s): A burst of requests for a 3D reconstruct service occurs (pressPedal event; purple, blue, olive). These requests then compete for transfer bandwidth and then, after transfer, are reconstructed sequentially, leading to a long response time for the olive request.
- Results are obtained that improve insight, e.g., about bottlenecks, and about the order of magnitude of cloud resources needed for a certain workload.
- Simulation performance, especially for application at runtime, is potentially a problem. Abstractions can deal with this, and we need to investigate cloud-specific or even application-specific modelling patterns to achieve this.
- Simulation-based analysis has difficulty with finding long and thin tails in distributions. For example, if we require that 99,9% of the requests are handled within x seconds, then the simulation-based analysis may report that this is true in 100% of the cases, whereas we know that there are situations in which the response-time threshold is violated. These just are not discovered by simulation. How to deal with this is an open question.

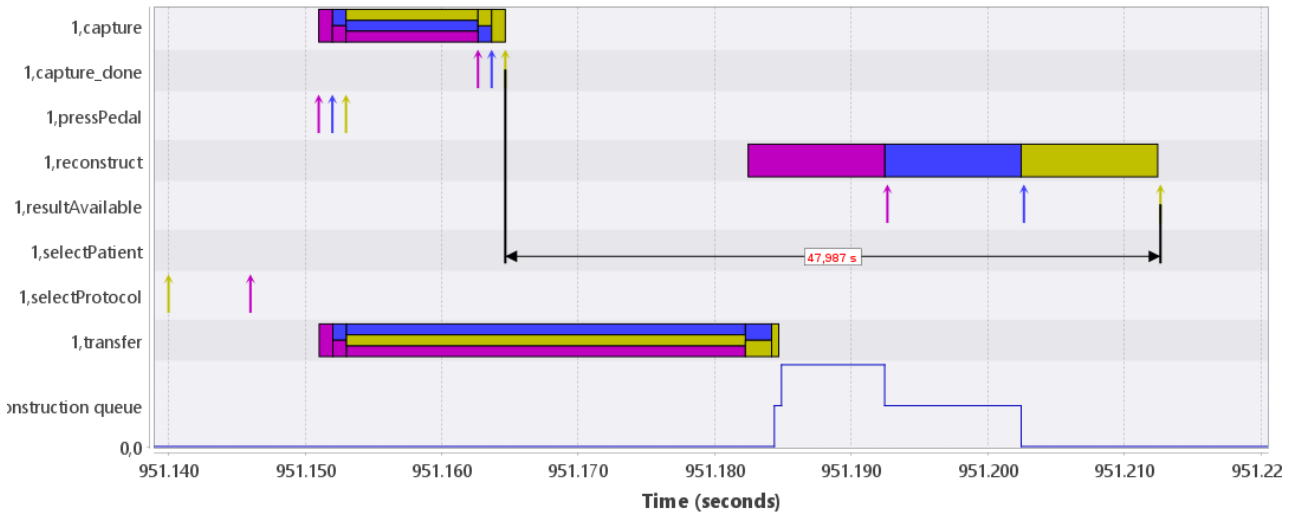


Figure 28: A long response time due to local bandwidth sharing and queueing in the cloud.

4.2.2.3 Performance analysis using formal methods

4.2.2.3.1 Overview

This section introduces some of our solutions for the Concept of Performance Modelling and Prediction, introduced in Section 6.4.2 of Deliverable D3.1. Our work specifically targets real-time applications with hard or soft timing constraints that execute on device, edge, or cloud continuum. In the context of TRANSACT project, examples of such applications can be found in UC4 (the 3D reconstruction application), and UC1 (the autonomous driving application).

We provide a response-time analysis method whose goal is to derive sound bounds on the worst-case end-to-end response time of real-time applications running on device, edge, or cloud continuum. Similar to the POOSL-based simulation technique introduced in the previous section, our method is used in the “Safety, Performance, and Security Monitoring Services” and “Operational Mode Manager/Coordinator” of either of the three tiers (Device, Edge, and Cloud) of the TRANSACT reference architecture (see Figure 29).

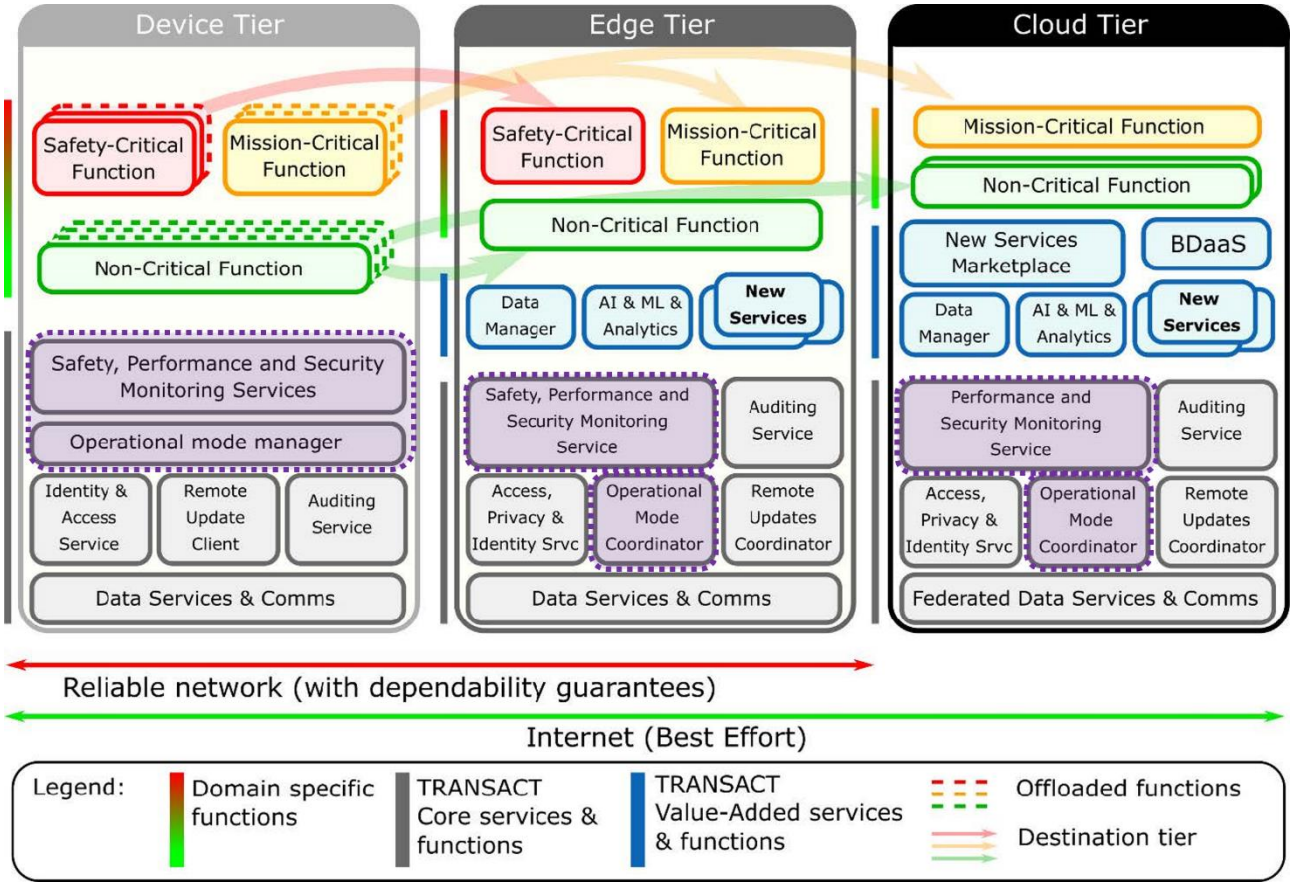


Figure 29: The position of our formal-method-based performance analysis in the reference architecture.

Our method can be applied both at design time and at run time throughout the system's lifecycle (see Figure 25). From the moment that the timing model of the application (i.e., the components of the application and their interactions with each other) is known and some early results on the worst-case execution time (WCET) of each part of the application is available, our analysis can be used to determine the worst-case response-time of the application (executed on the device, edge, or cloud continuum). In later stages, our analysis method can be used during the execution of the system to check whether new configurations of the application or the underlying computing resources can still satisfy the timing constraints of the system. The only remark is that the runtime of our analysis is in the order of minutes and therefore it must be used well before the planned reconfiguration can be applied on the system.

In this deliverable, we present the state of the art on the worst-case response-time analysis (WCRT), introduce our solution, and explain how it advances the state of the art. Later in the deliverable D3.5 (that is due in M33), we will work on refining our methods and reducing their run time by using partial-order reduction.

4.2.2.3.2 State of the art

This section is organized in four parts: (i) the worst-case response-time (WCRT) analysis problem and its hardness, (ii) the state of the art on classic critical-instant-based WCRT analyses methods, (iii) a brief review of the model-checking based methods for WCRT analysis problem, and (iv) the reachability-based WCRT analysis using schedule-abstraction graph (SAG) method. Our contribution in this project is to extend the fourth category of response-time analyses to support edge- and cloud-based real-time applications.

The worst-case response-time analysis problem

Deriving the WCRT of a system (and its components) is one of the crucial steps towards guaranteeing predictable timing performance (or timing behaviour), and therefore, the temporal correctness of real-time systems. In a nutshell, this problem can be described as follows (Buttazzo G. C., 2011):

> ***The WCRT analysis problem.*** *Given a set of real-time and non-real-time applications (tasks) that are executed on a computing platform, and a scheduling (or a resource-management) policy that decides which task must be executed at what time on which resource, determine the worst-case response time that each of these applications may observe during its lifetime.*

This problem is known to be NP-Complete (Eisenbrand, EDF-schedulability of synchronous periodic task systems is coNP-hard, 2010) (Eisenbrand, 2008) even under simple system configurations, namely, when the system runs a set of periodic tasks (i.e., tasks that are activated periodically by a timer interrupt) on a single-core hardware platform using a fixed-priority scheduling policy (Eisenbrand, 2008). Therefore, intuitively, the WCRT problem for distributed (e.g., microservice-based) applications that execute over multiple computing resources (device, edge, and cloud continuum) is also at least an NP-Complete problem since it can only be harder (in terms of complexity) than the response-time analysis for systems with a single computing resource.

Critical-instant based WCRT analysis methods

The WCRT analysis problem has been the focus of real-time systems community since half-a-century ago (Liu, 1973). One of the most prominent approaches to this problem is based on a concept called the 'critical-instant theory'. In a nutshell, the critical-instant theory defines the worst-case *arrival pattern* of a set of recurrent tasks such that under this arrival pattern, the real-time task (that is under analysis) has the largest response time in its lifetime (Audsley, Burns, Richardson, Tindell, & Wellings, 1993). An upper bound on such a response time is then obtained using a fixed-point equation that converges in a pseudo-polynomial time.

Over years, the critical-instant based analysis has been extended to various system types from single-core platforms (Audsley, Burns, Richardson, Tindell, & Wellings, 1993) (Davis, Burns, Bril, & Lukkien, 2007) to multi-core platforms (Baker, 2003), and from single-threaded applications (with sequential codes) to multi-threaded applications (parallel tasks) with dataflow or control-flow dependencies (Melani, Bertogna, Bonifaci, Marchetti-Spaccamela, & Buttazzo, 2015) (Serrano, Melani, Bertogna, & Quiñones, 2016) (Casini, Biondi, Nelissen, & Buttazzo, 2018). It has also been extended to distributed systems in which deriving an upper bound on the end-to-end response-time, latency, and *data age* are of concern (Feiertag, Richter, Nordlander, & Jonsson, 2009). Data age is a relevant performance metric for real-time distributed applications where the time interval from the moment that an input data has been acquired from the environment to the last moment at which the system actuates upon that input data must be bounded.

In the context of edge-based or could-based applications, the most relevant response-time analyses are those designed for (i) parallel applications (Casini, Biondi, Nelissen, & Buttazzo, 2018) and (ii) *gang tasks* (Feitelson & Rudolph, 1992) that run on powerful multi-core processors. A *gang task* is a multi-threaded task that requires certain number of cores before it can start its execution because the threads must execute concurrently and/or synchronously on a typically large input data. These two classes of task models encompass most common types of microservice applications, AI-based applications (such as deep neural networks), and applications with heavy numerical computations.

A common drawback of the WCRT analysis methods that are based on the critical instant theory is that they result in an overly pessimistic response-time bounds (see a comparison in (Nasri, Nelissen, & Brandenburg, 2018) (Nasri, Nelissen, & Brandenburg, 2019) (Nasri & Brandenburg, 2017) (Yalcinkaya, Nasri, & Brandenburg, 2019)). The reason is that as the system (or application) becomes more complex, it becomes harder to find a realistic worst-case arrival pattern, and hence, a realistic upper bound on the actual WCRT of the tasks. Consequently, when used during the design (or run time) phase, these methods typically lead to having a

worse average-case timing performance and low resource efficiency (namely, the computational resources are left idle most of the time).

Model-checking based WCRT analyses

Model checking has been used to derive accurate bounds on WCRT of a set of tasks scheduled by a fixed-priority scheduling policy on a single-core (Baker & Cirinei, 2007) or multi-core platform (Guan, Gu, Deng, Gao, & Yu, 2007). Baker and Cirinei (Baker & Cirinei, 2007) proposed a WCRT analysis that uses a finite state machine to models all possible combinations of arrival times and execution sequences of the tasks in the system. This method was later improved by (Burmyakov, Bini, & Tovar, 2015) (Burmyakov, Bini, & Lee, 2022), however, due to its inherent exponential state-space exploration, none of these methods can be used on an actual edge- and cloud-based use case. This drawback also persists in model-checking approaches such as Timed Automata (TA) (Yalcinkaya, Nasri, & Brandenburg, 2019) and hybrid automata (Sun & Lipari, 2016). A comparison in (Yalcinkaya, Nasri, & Brandenburg, 2019) shows that these methods do not scale for any reasonable system configuration for edge- and cloud-based applications (that has, e.g., 4 cores). Moreover, there is no such analysis for multi-threaded applications which further limits the applicability of these works in the context of TRANSACT project and its use cases.

Reachability-based response-time analysis using schedule-abstraction graph (SAG) method

Recently, a new approach to the WCRT analysis problem, has been introduced that is based on *reachability analysis*, a concept originally introduced in 1978 for the analysis and verification of communication protocols in distributed system. This method is called the *schedule-abstraction graph* (Nasri & Brandenburg, 2017) (Nasri, Nelissen, & Brandenburg, 2018) (Nasri, Nelissen, & Brandenburg, 2019).

In a nutshell, a reachability based WCRT analysis method automatically explores the space of possible schedules (execution sequences) of tasks/jobs in the system without incorporating the impossible schedules (those that can never occur under the given assumptions). As a result, it does not rely on defining the worst-case arrival pattern of the tasks, instead, it explores all task arrival times and their respective response times that could potentially happen during the system's lifetime. This leads to having far more accurate (tight) bounds on the response-time values.

The SAG method (shown in Figure 30) takes as input a timing model of a workload (for example, a set of jobs in which each job is characterized by its arrival time, execution time, and deadline), a model of the resources in the system (for example, the number of cores available on the computing platform), and the scheduling policy (for example, fixed-priority scheduling policy, FIFO, EDF, etc.). It finds the worst-case and best-case response time of each of the items (jobs) in the input workload model by constructing a (directed) graph that abstracts all possible schedules that can happen when the input workload model executed on the resources of the platform and is scheduled by the given scheduling policy.

In the schedule-abstraction graph, each vertex represents a system state, where the state contains information about the status or availability of each of the computing resources. This information is abstracted as an uncertainty interval. An uncertainty interval $[a, b]$ for a resource R means that the resource is not available to be used by any new job/task before time a, it may become available at any time from a to b non-deterministically, and it will certainly be available after time b. This abstraction allows combining many states into one. An edge from state v_p to a state v_q in the schedule-abstraction graph is labelled with the decision of the scheduler. It shows which job has been dispatched after the state v_p and has evolved this state into a new state v_q.

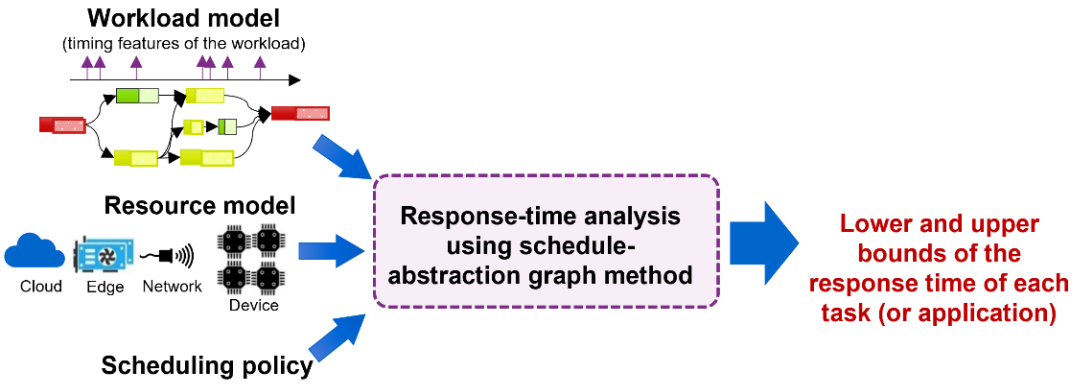
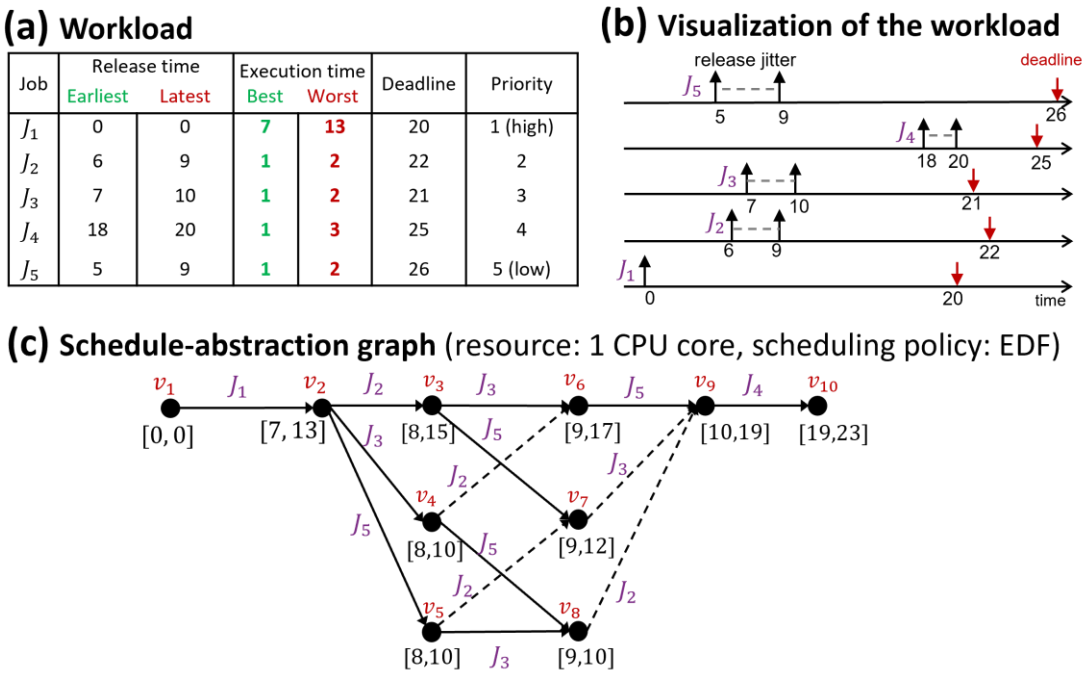


Figure 30: SAG method for response-time analysis

Every path from the initial state to the sink state in the schedule-abstraction graph represents all possible schedules of the entire workload in which the schedules have the same order of execution between the jobs in the input workload. Different paths demonstrate schedules that have a different job ordering. SAG method builds the graph using a breadth-first approach, namely, it starts from an initial state in which the system is idle. It adds one outgoing edge to this state for every job that can be dispatched first on the platform (given all possible arrival scenarios). Then for each newly created state, it repeats the same process until each path from the initial state to a leaf state has as many edges as the number of jobs in the job set. In order to reduce the size of the graph, the SAG method merges paths that contain the same set of jobs.

Figure 31 shows an example workload with 5 jobs (each job is identified by its earliest and latest arrival times, best-case and worst-case execution times, deadline, and priority) to be scheduled by the earliest-deadline-first (EDF) scheduling policy on a single core platform. In this example, the resulting SAG shows that when jobs $\{J_1, J_2, J_3\}$ are dispatched one after each other, the core might be available at earliest at time 9 and at latest at time 17 (shown in Figure 31-(c) state v_6).

To obtain the response-time of each job, SAG stores the earliest and latest finish time of each job that appears on each edge. Hence, for each job the best- and worst-case response times are the minimum and maximum observed finish time in the graph.



Job	Release time Earliest	Release time Latest	Execution time Best	Execution time Worst	Deadline	Priority
J_1	0	0	7	13	20	1 (high)
J_2	6	9	1	2	22	2
J_3	7	10	1	2	21	3
J_4	18	20	1	3	25	4
J_5	5	9	1	2	26	5 (low)

Figure 31: An example of SAG

Limitations of SAG

Despite being an efficient and accurate response-time analysis, the SAG method still suffers from state-space exploration when the arrival time of the jobs in the input workload have large uncertainty (jitter). In the context of the TRANSACT project, we will try to address this limitation by incorporating partial-order reduction rules. This result will be part of the Deliverable D3.5.

The second limitation of SAG is that it does not support gang-task model and hence cannot model the execution of applications on a Kubernetes-Pod that has multiple CPU cores. When an application is sent for execution on a Pod, depending on the number of available cores, it may claim one or more cores. However, in the current SAG method, a job can claim only 1 core at a time. We will address this limitation in the following subsection of the report.

4.2.2.3.3 Innovation step and solution description

Inspired from the use cases UC1 (autonomous vehicles) and UC4 (image-guided therapy and the 3D reconstruction of MRI images on the cloud) in the TRANSACT project, we consider extending the state of the art on the response-time analysis for edge- and cloud-based applications in two ways.

Innovation 1: a new WCRT analysis method for edge- and cloud-based applications that use a gang model

This innovation targets applications such as 3D reconstruction (for UC4) that require accessing multiple processing cores at the same time before they can start their execution (namely, they have a gang-task model). The goal of this innovation is to derive response time bounds of such applications. As stated before, there is no research work in the state of the art to solve this problem for applications that can execute (without interruption) on the resources assigned to a container (for example, a *Kubernetes Pod*).

Figure 32 summarizes the three types of gang tasks: rigid, moldable, and malleable. Rigid gang tasks always claim a fixed number of researches, and they cannot start their execution if those resources are not available. Moldable gang tasks can execute on any number of resources from a given minimum to a given maximum

value. For example, if only one core is available, they start executing on that core, but if two cores are available, they claim both cores and start executing on the two of them. Malleable gang tasks are moldable gang tasks that can increase the number of resources they claim as soon as more resources become available during their execution. Due to implementation complexity, applications typically use rigid or moldable gang task model and hence we also focus on these two.

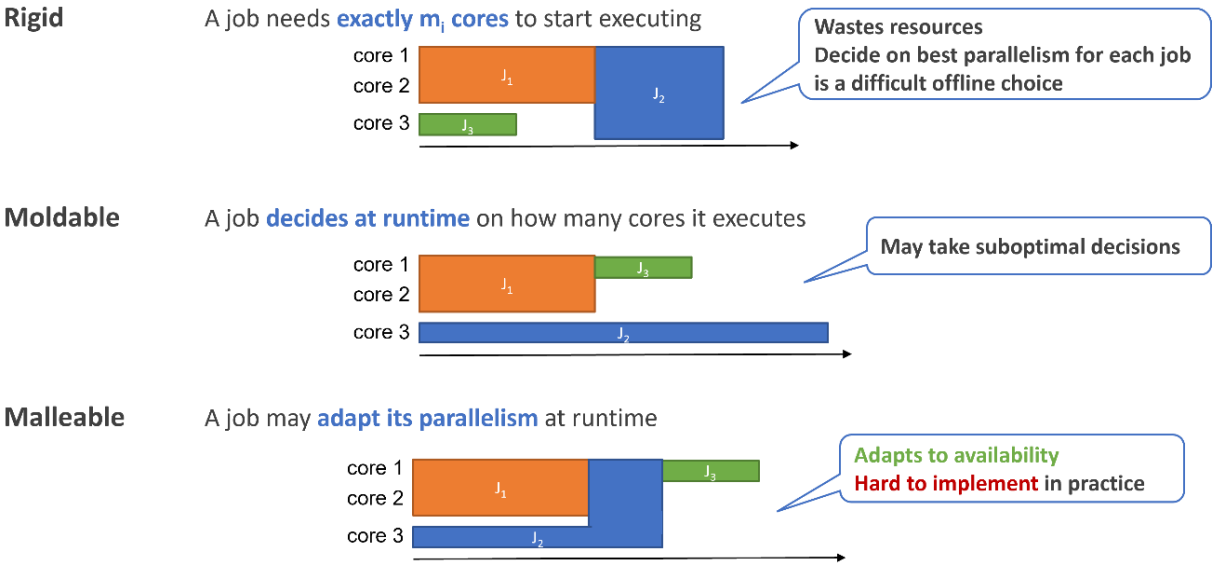


Figure 32: Types of gang tasks

We extend the schedule-abstraction graph (SAG) method which is currently the best-performing response-time analysis in terms of accuracy and scalability, to derive the upper bound on the WCRT of the applications that follow a rigid or moldable gang model. This extension, which has been published at one of the top-tier conferences on real-time systems (Nelissen, Marce-i Igual, & Nasri, 2022), requires three steps: (i) formalizing the scheduling policy that can be used to execute gang tasks on a Pod (that owns a number of cores), (ii) defining a new SAG method that can account for varying number of requested cores for each job of a gang task, and (iii) proving that the new method provides correct (sound) upper bounds on the WCRT of such tasks.

We use one of the available scheduling policies in Linux, namely, the fixed-priority scheduling policy, as it is one of the most widely used scheduling policy for real-time applications and is supported by a wide variety of embedded and non-embedded operating systems such as Linux, Pike-OS, FreeRTOS, etc. (Akesson, Nasri, Nelissen, Altmeyer, & Davis, 2020). We consider that the scheduling policy assigns the maximum number of currently available cores to a gang task (provided that the number of cores that the task can concurrently occupy is no larger than the number of assigned cores).

To account for the varying number of available cores (at run time), we redesign SAG method as follows. We annotate the edges of the SAG by not only mentioning which job is to be dispatched next, but also on how many cores. For example, if a system state v_p allows a job to be dispatched on 1 or 2 cores (in different scenarios), then there will be two outgoing edges from the state v_p, as shown in the schematic example in Figure 33. In this figure, in state v_p, two jobs J_1 and J_2 are executing on the system, where J_1 occupies 1 core (at least until time 5 and at most until time 10) and J_2 occupies 3 cores (at least until time 10 and at most until time 15). If jobs J_1 and J_2 complete at time 10, then the not-yet-dispatched job J_3 is scheduled immediately at time 10, and it claims the maximum number of cores that it requires (here it needs at most 2 cores). This scenario is shown in the state v_g. However, if job J_1 finishes earlier than time 10 or J_2 does not

finish at the same time as J_1, then there will be only 1 available core on the platform, which will be immediately claimed by J_3. This scenario evolves the state v_p into the state v_q.

In order to automatically capture all possible scheduling scenarios that can evolve a given system state, we develop new expansion rules for our new SAG method. To defer the state space explosion, we also introduce new merging rules to combine similar states. Details of these steps and their proof of correctness can be found in our paper (Nelissen, Marce-i Igual, & Nasri, 2022).

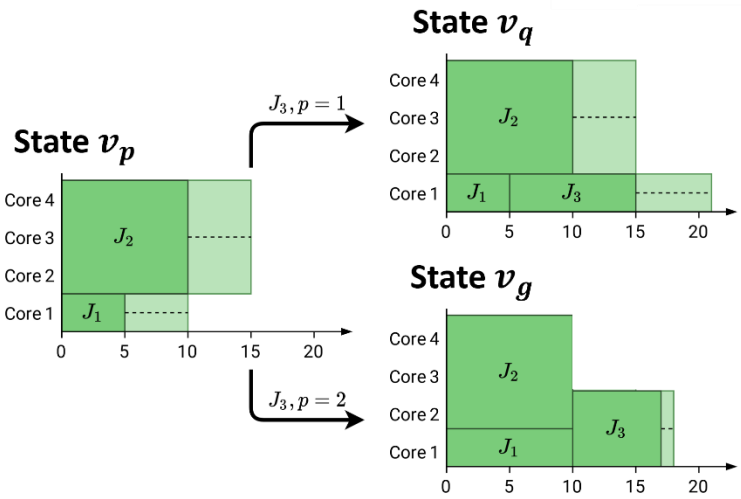


Figure 33: A schematic example of our extension to SAG (for supporting gang tasks)

Innovation 2: a new end-to-end latency analysis for autonomous driving applications (for UC1)

The goal of this innovation is to obtain the worst-case *data-age* (which is one of the critical latency metrics for automotive systems) for autonomous driving applications (such as the one in UC1). Data age is defined as the longest time between the moment that a sensor senses a data (e.g., a camera captures an image of a pedestrian that approaches the car) to the moment that the system still reacts based on that data (e.g., until the moment that the system still assumes that there is a pedestrian near the car) (Feiertag, Richter, Nordlander, & Jonsson, 2009). For a real-time system, it is crucial to have a realistic view of the environment because '*reacting based on an old data*' can be as dangerous as '*not reacting to a new data*'.

In autonomous driving systems, many tasks are involved in a chain of actions that allow the system to react to its surrounding environment. For example, the sensor data (from camera, lidar, radar, etc.) is first pre-processed (by removing possible noises, locating the object, identifying the object) and then the system takes a decision based on that data (e.g., by deciding to stop the car in very hazardous situations, speeding up or changing the course to avoid clashes, or doing nothing), and finally this decision is applied on different actuators all around the car (for example, to stop the car, four wheels must receive and apply the break command almost at the same time). Consequently, obtaining the worst-case data age requires (i) analysing the response-time of various tasks in a task chain, where each task could be executed on a distributed computing hardware (that are spread across the car's body, or placed on the edge or cloud), and (ii) analysing the relation between the start time of each data consumer task and the completion time of a data producer task in the task chain. Figure 34: An example of an automotive application, its tasks (and their periods), and its task chainsFigure 34 shows an example of an autonomous driving application which consists of 8 periodic tasks (Hamann, et al., 2019).

To ease applicability of task chain model to distributed asynchronous systems (like the ones used in a car), the automotive standards (Fürst, et al., 2009) typically use a non-binding data flow relation between tasks,

namely, whenever a data-producer task completes, it copies its output (data) into a global shared space (or buffer). Similarly, every time a data-consumer task start executing, it accesses the global shared space and reads the current value of data that is stored in that space. This model, which is called implicit data communication, allows the data consumer task to execute independently from the data producer task and hence it relaxes the need for synchronization between these tasks.

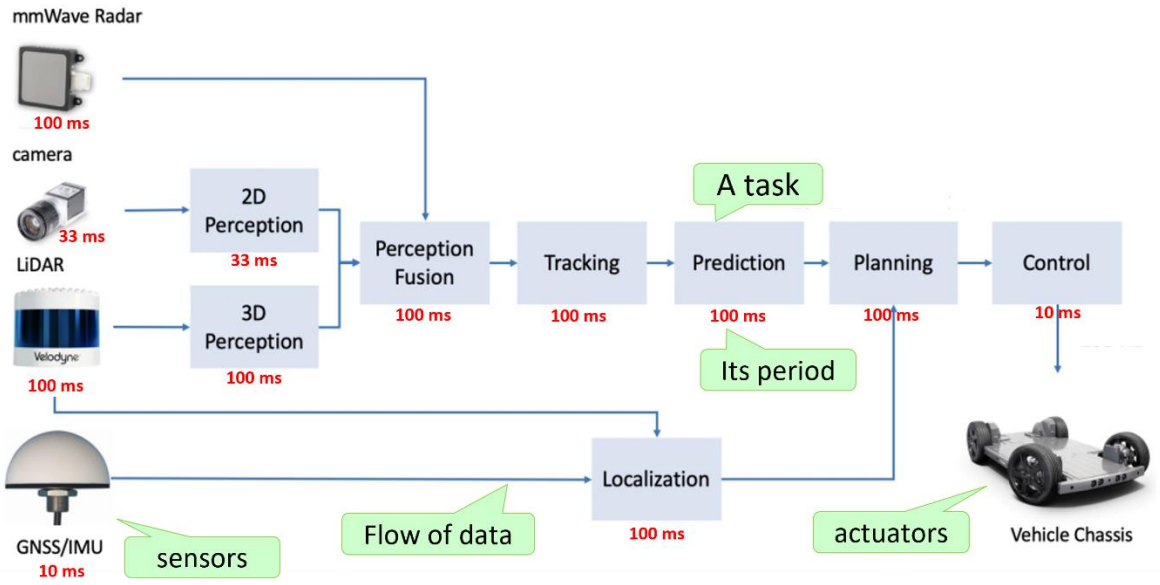


Figure 34: An example of an automotive application, its tasks (and their periods), and its task chains

In general, a task system might have more than one possible schedule due to the fact that the execution time of the tasks may vary at runtime (e.g., due to taking different paths in the program, input data variations, context switch overheads, etc.). Similarly, the time at which a task becomes ready to be executed (a.k.a. release time) typically has jitter (caused by the overheads of the interrupt service routines in the underlying operating system, network and queuing delays, etc.). Figure 35 (a) and (b) show a task set with two different schedules in which the age of data used by the task chain (Task 1, Task 2, and Task 3) depends on the scenario. In this example, we assume that the hardware platform has only one processor.

Obtaining the worst-case data age for a system requires exploring all possible scenarios (schedules) that can happen for the system during its lifetime. Since there can be exponentially many schedules, enumerating them all naively is not practical. To perform this exploration more efficiently, we use the schedule-abstraction graph method as a basis to derive the *earliest and latest start time* (EST and LST) and *earliest and latest finish time* (EFT and LFT) of each instance (a.k.a. job) of a given periodic task set over an interval of time called hyperperiod in which the arrival pattern of periodic tasks repeats. Hyperperiod of a set of periodic tasks is equal to the least-common multiple of their periods (Buttazzo G. C., 2011). For the example shown in Figure 35, the hyperperiod is 60 (since periods are 10, 20, 30, and 60).

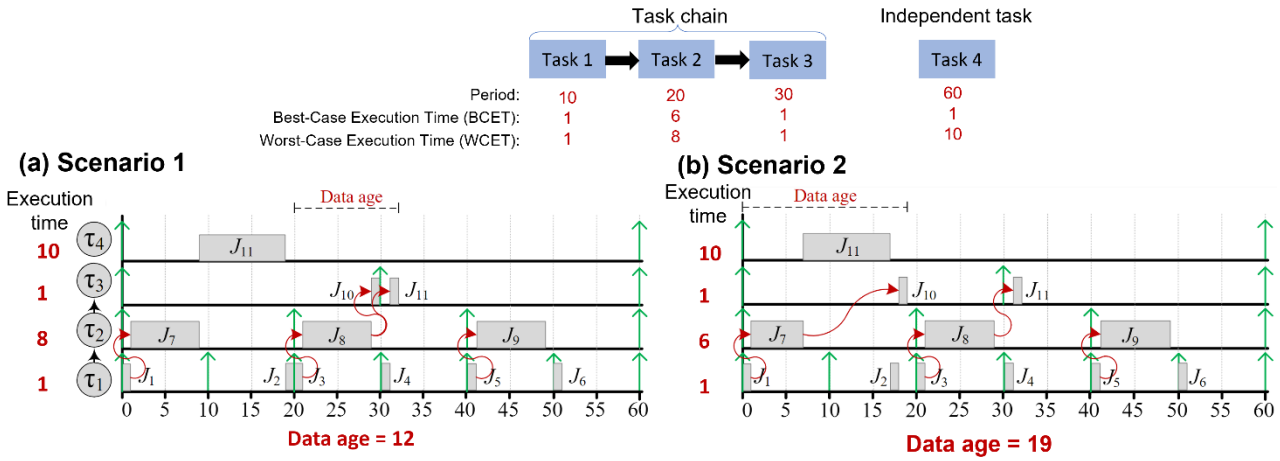


Figure 35: An example of a task chain with three tasks and two possible schedules

Figure 36 shows the overall view of our solution to derive the upper and lower bounds of data age. Obtaining the earliest and latest start time and finish time of the jobs (instances of tasks in a hyperperiod) alone is only the first step towards deriving the bounds on data age because depending on the actual start time and finish time of the jobs, two jobs may or may not be the data consumer and data producer for each other. In the example shown in Figure 37, only a subset of jobs of Task 2 can be data producers for the job of Task 3, namely, J_7,J_8, and J_9. The reason is that the output of the jobs J_5 and J_6 will be overridden by themselves or by J_7 before J_{11} (the data consumer job) can have any chance to start its execution. Similarly, since J_{11} reads its input data only during the interval [EST, LST], there is no way that it can read a data that is produced by job J_{10} since the EFT of J_{10} is larger than the LST of J_{11}.

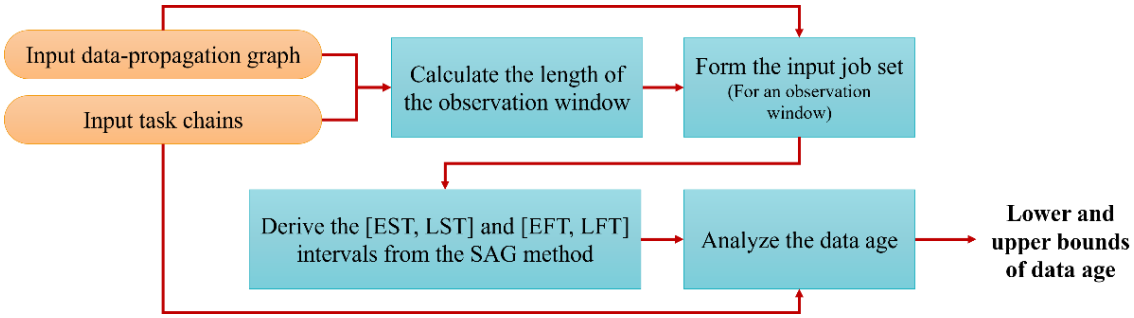


Figure 36: Overall steps of our solution to analyze data age

In our paper (Gohari, Nasri, & Voeten, 2022), we propose a method to find the set of potential data-producer jobs for a data consumer job quickly and accurately. We use this method to find the data age of each chain of jobs within one hyperperiod and then derive the best- and worst-case data age of each task chain. Details of our solution can be found in the paper.

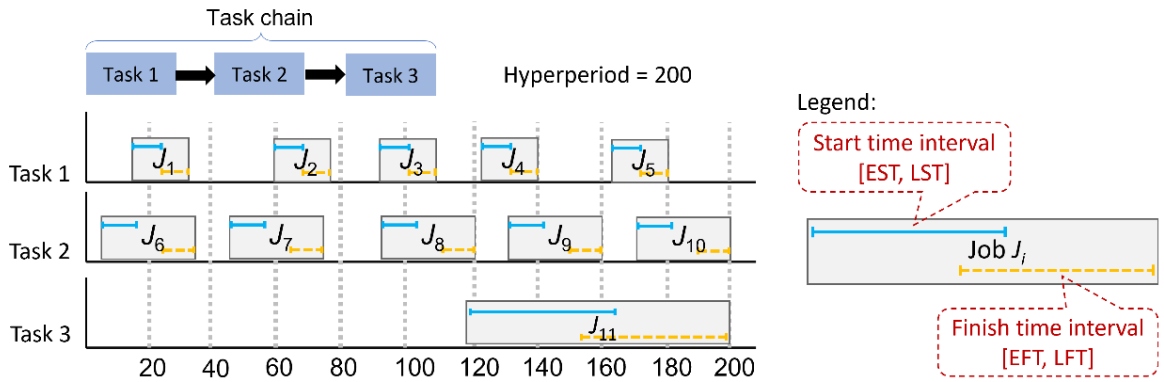


Figure 37: An example output from the SAG method for a task set with 3 tasks

4.2.2.3.4 Application to use case(s)

As mentioned in the introduction, our work focuses on real-time applications with hard or soft timing constraints that execute on device, edge, or cloud continuum. In particular, we consider UC4 (the 3D reconstruction application) and UC1 (the autonomous driving application). In the following we evaluate our innovations (introduced in the previous subsection) using case studies and synthetic workloads.

Evaluation of our response-time analysis for gang tasks

As explained earlier, TUE has an in-house demonstrator that mimics the 3D reconstruction application used in UC4. This application has been implemented as a single-thread code that runs in a Kubernetes Pod in our initial demonstrator. As a result, the current implementation cannot take advantage of the available cores on a Pod and hence, cannot be used as a case study for our response-time analysis of gang tasks at the moment. Therefore, we use synthetic workload models (applications that are generated randomly) to evaluate our solution.

For the following experiment, we considered multi-threaded rigid gang tasks that are activated periodically. We assess the impact of the system workload (a.k.a. 'utilization') on the acceptance ratio of our method. The system utilization represents the amount of work that is released to the system per unit of time and the acceptance ratio represents the number of systems whose timing requirements is satisfied divided by all systems considered. A system is accepted by our method if the WCRT of each task obtained by our method is smaller than the deadline of that task. We generated 200 task sets per utilization value (hence, in total, 1800 randomly generated task sets) and showed the average acceptance ratio of our analysis on the vertical axis of Figure 38. Details of our task set generation method have been sketched in our paper (Nelissen, Marce-i Igual, & Nasri, 2022).

Figure 38 shows that our solution significantly improves the accuracy of the response-time analysis for gang tasks in comparison to the state of the art (Dong & Liu, 2019). Moreover, we see that as the utilization increases, the system becomes busier and hence tasks began to have larger response-time than their deadline. This results in a decrease in the acceptance ratio of our method at higher utilizations. We have also evaluated the impact of the maximum number of cores that the tasks in a task set may request. We observe that the best configuration is when the gang tasks request for all available cores.

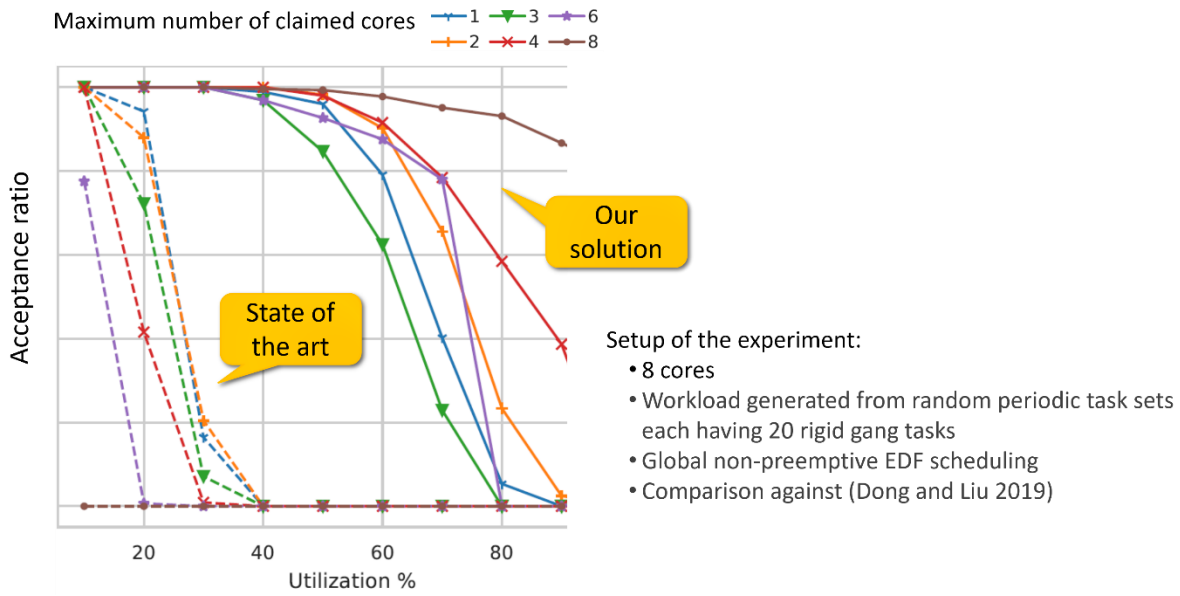


Figure 38: Experimental results of our method for gang tasks

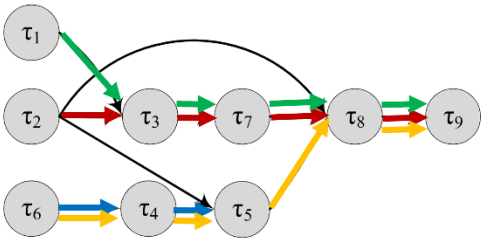
Evaluation of our data-age analysis for task chains in automotive applications

We have evaluated our solution using a case study from (Hamann, et al., 2019) as well as synthetically generated task sets. We considered the case study introduced by Bosch GmbH (Hamann, et al., 2019) which presents an autonomous driving vehicle with 9 periodic tasks and 4 task chains of interest shown in Figure 39. As shown in this figure, our method provides about 30% tighter bounds on the worst-case data age in comparison to the state of the art.

To evaluate the impact of task set utilization on our solution, in the next experiment we have generated random periodic task sets using benchmark tasks introduced by (Kramer, Ziegenbein, & Hamann, 2015) with varying utilization (from 10% to 90%) for a platform with 4 cores. The result of this experiment is shown in Figure 40 (a). Moreover, to evaluate the impact of the number of cores of the computing platform on our analysis, we also designed an experiment in which we kept the utilization constant (equal to 50%) and varied the number of cores (from 2 to 8), as shown in Figure 40 (b). In both experiments, we observe that our solution outperforms the state of the art in terms of the tightness of the derived bounds on the worst-case data age by about 30% on average.

Case study: Advanced driver-assistance system (Hamman et al. 2019, Verucchi et al. 2020)

Data dependency graph



Task chains with deadlines:

$\{\tau_6, \tau_4, \tau_5\}$
$\{\tau_1, \tau_3, \tau_7, \tau_8, \tau_9\}$
$\{\tau_2, \tau_3, \tau_7, \tau_8, \tau_9\}$
$\{\tau_6, \tau_4, \tau_5, \tau_8, \tau_9\}$

Task	Function	Description $(\sigma_i, [C_i^{min}, C_i^{max}], T_i, D_i, X_i)$
τ_1	GPS	$(0, [5, 7], 50, 50, \phi_1)$
τ_2	Lidar	$(0, [10, 12], 50, 50, \phi_1)$
τ_3	Localization	$(0, [22, 28], 50, 50, \phi_1)$
τ_4	Detection	$(0, [25, 28], 50, 50, \phi_4)$
τ_5	Fusion	$(0, [18.9, 25], 50, 50, \phi_2)$
τ_6	Camera	$(0, [1.8, 2], 25, 25, \phi_4)$
τ_7	EKF	$(0, [3, 6.5], 25, 25, \phi_5)$
τ_8	Planner	$(0, [3.2, 5], 10, 10, \phi_6)$
τ_9	Control	$(0, [1.8, 4.5], 10, 10, \phi_3)$

Task chain	Our solution	Becker et al. 2016 (schedule agnostic)	Becker et al. 2016 (using WCRT)	Improvement (%) (w.r.t. Becker et al. 2016)
Task chain 1: $\langle \tau_6, \tau_4, \tau_5 \rangle$	[68.9, 75]	[75, 125]	[75, 100]	25%
Task chain 2: $\langle \tau_1, \tau_3, \tau_7, \tau_8, \tau_9 \rangle$	[61.8, 114.5]	[59.5, 190]	[64.5, 164.5]	30.3%
Task chain 3: $\langle \tau_2, \tau_3, \tau_7, \tau_8, \tau_9 \rangle$	[61.8, 114.5]	[59.5, 190]	[64.5, 164.5]	30.3%
Task chain 4: $\langle \tau_6, \tau_4, \tau_5, \tau_8, \tau_9 \rangle$	[81.8, 134.5]	[84.5, 185]	[94.5, 179.5]	25%

On average, we provide 28% tighter upper bounds in comparison with the state of the art

Figure 39: Evaluation of our data-age analysis method on a case study

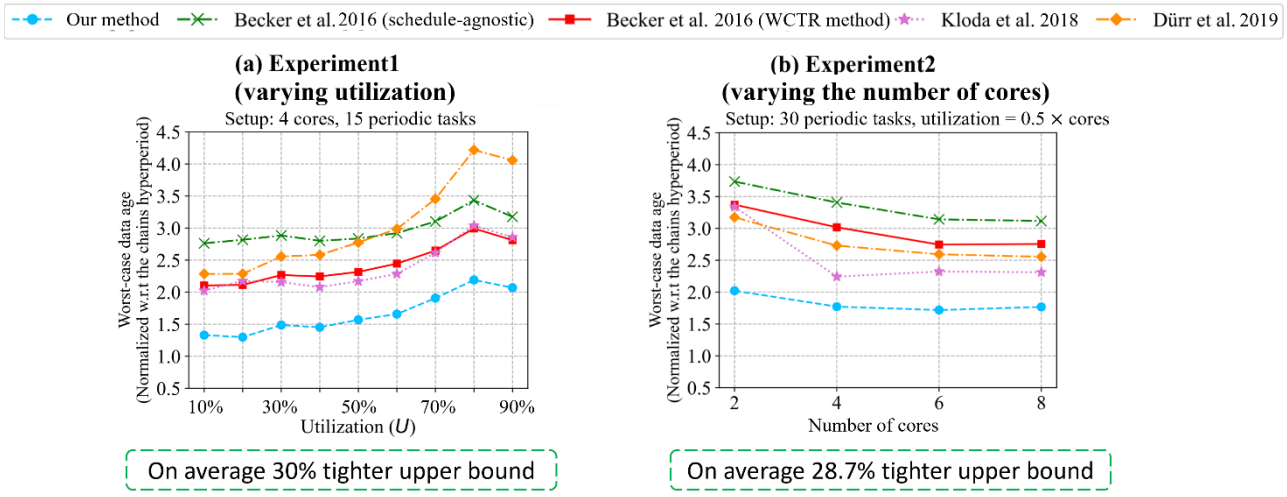


Figure 40: Evaluation of our data-age analysis method on synthetic task sets

4.2.2.4 Workflow simulation

4.2.2.4.1 Overview

Today, systems for image guided therapy (IGT) are tightly coupled integrating data acquisition, image formation and control into one system. Dedicated applications such as 3D reconstruction (3DRX) or advanced AI solutions are implemented on workstations installed on premise and are highly application/product specific. In UC4 an edge-cloud-based clinical applications platform for IGT systems is investigated virtualizing software components from the system hardware according to the TRANSACT reference architecture.

Other clinical imaging systems, such as computed tomography (CT) and magnetic resonance imaging (MRI), are also monolithic, self-contained systems. The image reconstruction of the acquired data is part of the

imaging workflow and runs on dedicated hardware part of the scanning device. Generally, the operator in the scanning room checks the quality of the examination based on the reconstructed images. Later, the radiologist uses the same images for reporting. Scanner manufacturers must therefore invest into the optimization of the algorithms balancing cost for development and the bill of materials while the customers must balance patient throughput and quality of the images. New, emerging algorithms, such as 3D, model-based or AI-based methods do not allow to interleave acquisition and reconstruction and/or need significantly longer compute times, may not be accepted in clinical practice (as a simple replacement of the conventional algorithms) because they will block further scans of the patient, resulting in longer examination times.

Therefore, the edge/cloud solutions and their analysis with respect of performance and security considerations investigated in UC4 for IGT are also interesting for these medical imaging devices. Specifically, for MRI the image reconstruction can be modularized, and the reconstruction software moved to other locations (on-premise, cloud) as illustrated in Figure 41. Please note, that in contrast to IGT, no patient interventions are preformed, so requirements on patient safety and degradation strategies are less demanding for MRI.

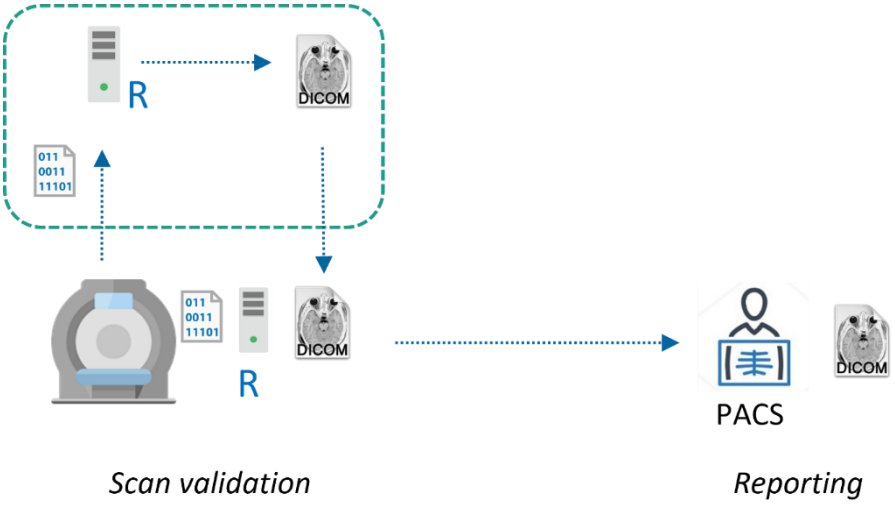


Figure 41: The MRI reconstruction is not moved to another location.

In Figure 41, the reconstruction is still coupled to the data acquisition and has to be as fast as possible to avoid longer patient examination times in the workflow. The required transfer of the data adds a time penalty to the total reconstruction time but on the other hand more powerful compute resources might allow for faster compute times.

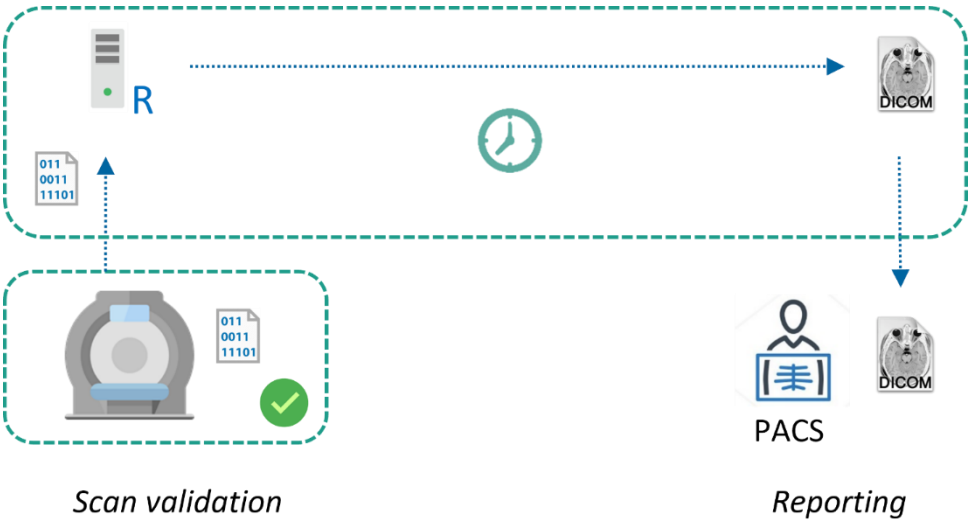


Figure 42: The MRI reconstruction is executed at a different point in time.

Figure 42 shows a different scenario, where the reconstruction is no longer part of the patient workflow. The operator will only receive an indication whether the acquired data is valid for reconstruction but the actual reconstruction is delayed but still guaranteeing that the images are reconstructed when the radiologist plans for reporting. The so-called report turnaround time (RTAT) in hospitals depends on the type of examination, but studies indicate average RTATs for MRI on the order of hours (Netsch, 2019).

In this task we investigate how the scanner and reporting workflow is changing when the reconstruction is executed at a different location or point in time. We use information from the machine log files such as the timestamps of various request and other information such as scan names and their parameters over a period of time to derive an exact scanning workflow. Based on this data we will model different reconstruction scenarios, for example, a scenario where time-consuming reconstructions are transferred to the hospital's IT department using different bandwidth constraints and compute, or simulate the effect (what-if) of this on the scanner workflow, such as the delays of patient examinations.

This work thus extends in some aspects the work from section 4.2.2.2 in that it considers not only the CPS and its parameters, but also the department operating the CPS (a medical device) and how the change of the CPS parameters would affect the workflow. The results may give an indication on parameter ranges for the CPS so that the effect on the department workflow is minimal, or alternatively might suggest alternative workflows that maintain or even improve certain KPIs on the departmental level.

4.2.2.4.2 State of the art

Discrete event simulation (DES) is an established method to model health-care systems for many years, specifically for optimization of workflows in the area of emergency care. Good literature overviews are from 2010 and 2012, respectively (Günal, 2010) and (Mielczarek, 2012). In a more recent application Ranschaert et al investigate in a recent publication the potential of artificial intelligence (AI) in radiology beyond image analysis. AI can be used to optimize all steps of the radiology workflow by supporting a variety of non-diagnostic tasks, including order entry support, patient scheduling, resource allocation, and improving the radiologist's workflow (Ranschaert, 2021).

For MRI, specifically MRI reconstruction, there is not much literature published. Gun et al describe a method (not based on simulation) to improve the MRI scanner utilization using modality log files (Gunn, 2017). Frydrychowicz at al derive statistical data of scans from MRI log files but only visualize the data by using dashboards (Frydrychowicz, 2021).

4.2.2.4.3 Innovation step

Allowing more flexible reconstruction scenarios provide many new and innovative aspects:

- The data for the simulations do not build on general figures and assumptions about MRI. We are using data from clinical routine of a radiology department of a hospital. Such data can also be used to derive realistic models of MR scanning which are less sensitive to privacy and other regulations. This allows us to generate what-if scenarios using data of a true workday in a hospital.
- Radiology departments usually optimize their workflow according to certain key performance indicators (KPIs) such as the average patient examination time and number of patient rescans. We can include this into our simulations and find scenarios for reconstruction which optimize a certain KPI. In such cases we simulate many scenarios and find the ones which have the best effect on the selected KPI.
- For a concrete product, such as a cloud-based reconstruction center offering hospitals additional reconstruction services, technical requirements such as transmission bandwidth needed and required compute resources and their scaling can be answered by simulations based on data from clinical routine before the service is being built. Also customer specific requirements can be taken into account, and the realistic prizing models derived.
- Today, reconstruction algorithms introduced into clinical practice (and to be accepted by radiologist) have to be very fast. For 2D reconstruction algorithms this is usually achieved by interleaving the data acquisition and reconstruction, to that the total waiting time for the operated is determined by the reconstruction time of a single slice, which is on the order of several seconds. This is not possible for emerging model-based 3D and 4D acquisition techniques which will provide better image quality. Furthermore, the reconstruction algorithms may need minutes which is not accepted by customers. We can, for example, exchange certain scans by a new class of reconstruction methods and simulate the effect on the workflow. Particularly, the RTAT window can be used as an additional simulation parameter and to demonstrate the impact on the workflow in radiology and to suggest solutions to the customers and test them for acceptance and practical considerations. This may also have a positive effect on research to further new methods without being limited by their run time.

4.2.2.4.4 Application to use case(s)

The work performed in this task complements UC4 in such a way, that we are including also other imaging devices, specifically MRI, and that we are investigating the changes in the patient and reading workflow, when moving system components, specifically the image reconstruction, into the cloud. We will also complement the simulations performed in UC4 on system and architectural level by simulations considering the effects on a work day of an entire department in a hospital.

For the simulation we may use straightforward computations but may also re-use and adapt the POOSL modelling developed in UC4 for more advanced scenarios. It is planned to use machine log file data from Karolinska Institute, Sweden, where the approval procedure in order to access their historic procedure data is already initiated. For the execution of the simulation work we tender a master student project at TUE Eindhoven, one of the partners of this task.

4.2.3 Performance management

4.2.3.1 Scenario-based performance management and reconfiguration

4.2.3.1.1 Overview

In this section, we present the current status on performance management of distributed safety-critical cyber-physical systems. This refines the cross-cutting concept for "Performance management" from deliverable D3.1. We sketch our view of how a component model and modelling language for quality and resource management can be used to model a multi-objective optimization problem that is at the core of performance management. This model-based technique is envisioned to be applied at run-time and depends on the performance monitoring and prediction techniques, which are used to provide input to the model. In this deliverable, we present preliminary modeling results, and sketch our view on performance management. In the second version of this deliverable, D3.5 that is due in M33, we refine this work.

4.2.3.1.2 State of the art

Performance management in cloud-based systems: In (Jiang, Lu, Zhang, & Long, 2013) an auto-scaling scheme for cost-latency trade-off under cost and SLA constraints is introduced. They predict future workload and use queueing theory for the relating the platform capacity with latency.

Modelling based on components is a well-known approach to tackle the increasing complexity of modern systems. Pre-defined components with well-defined behaviours and interfaces are used as interoperable building blocks. These components can be composed to create larger components with more complex functionality, and ultimately complete systems. Well-known component models in the software domain are the Component Object Model (COM) (Box, 1998), the Common Object Request Broker Architecture (CORBA) (Object Management Group, 2012), JavaBeans (Sun Microsystems, 1997), and Open Services Gateway Initiative (OSGi) (OSGi Alliance, 2018). These are all generally applicable and do not depend on the application domain.

Examples of component frameworks with tool support for cyber-physical systems are the Behavior Interaction Priorities (BIP) framework (Basu, Bozga, & Sifakis, 2006) and the Ptolemy framework (Eker, et al., 2003). Other conceptual component models are, for instance, the I/O automata (Lynch & Tuttle, 1987) and timed I/O automata (Kaynar, Lynch, Segala, & Vaandrager, 2003) frameworks, which provide formal notions of components, composition, and abstraction.

Interface models for components have been introduced by (de Alfaro & Henzinger, 2001). An interface model of a component specifies what the component expects of the environment and what it can provide. This supports compositional refinement and therefore component-based design. Using this principle, interface automata capture temporal assumptions about how a component is used and how it uses other components. The authors of, e.g., (Wandeler & Thiele, 2005), (Geilen, Tripakis, & Wiggers, 2011) use interface-based design concepts for real-time aspects of systems. Contract-based design (Sangiovanni-Vincentelli, Damm, & Passerone, 2012) uses similar ideas. It specifies contracts of components as a combination of assumptions that the component makes about the environment and the guarantees that it then can give.

(Hendriks, Geilen, Goossens, de Jong, & Basten, 2021) introduces a component interface model that focusses on quality and resource management aspects based on multi-objective optimization techniques captured in Pareto Algebra (Geilen, Basten, Theelen, & Otten, 2007).

Besides component models, a wide range of general-purpose languages and DSLs is in use, and is being developed, for systems engineering. The Systems Modeling Language (SysML, (Hause, 2006)) is a modelling language for systems engineering applications inspired by UML (Medvidovic, Rosenblum, Redmiles, & Robbins, 2002). It supports the specification, analysis, design, verification and validation of a broad range of (systems-of-)systems. It does, however, not provide specialized support for QRM. QRML (Berg, et al., 2020)

is a DSL that is inspired by such DSLs and builds on DSL technology to make it broadly accessible and ease integration with other methods, without losing its focus on QRM. A key feature of QRML is its connection to monitoring and visualization.

4.2.3.1.3 Innovation step

We have made initial steps to adopt the tooling of the Quality and Resource Management Language (QRML, (Berg, et al., 2020)) in the TRANSACT context by refining our view on performance management with QRML, and by making a high-level model of a part of UC4. The refined view is explained below, and the UC4 model is explained in the next section.

QRML supports a formal interface model for quality and resource management that emphasizes abstraction, separation of concerns and reuse (Hendriks, Geilen, Goossens, de Jong, & Basten, 2021). It has been developed in the FitOptiVis project (ECSEL Joint Undertaking, 2018-2021, https://fitoptivis.eu), and is available at https://qrml.org. The central concept in QRML is a *component,* which has an interface consisting of six parts: input, output, provided budget, required budget, qualities and parameters. A configuration of a component is an explicit valuation for each of these parts, and can be interpreted as a particular mode of operation with a certain quality of service and certain resource requirements. The component model supports the Y-chart decomposition of systems into application components, platform components, and a mapping between them (by connecting required budgets of the application components to matching provided budgets of the platform components). QRML models have a precise mathematical semantics as a mathematical constraint problem and they can be translated into the input language of constraint solvers, such as Z3 (de Moura & Bjørner, 2008). Practical modelling is supported by a domain-specific language (DSL). We envision that QRML is the top-level modelling technique for performance management:

- It makes explicit the key parameters and key performance indicators (*qualities* in QRML),
- provides a formal framework for trade-off analysis / multi-objective optimization, and
- acts as a formal framework that brings together analysis results from other performance monitoring and prediction tools (e.g., from AI-based performance prediction of Section 4.2.2.1, and from the simulation-based performance analysis of Section 4.2.2.2).

QRML-based techniques can be used in all tiers of the TRANSACT reference architecture and allows abstraction and clear separation of performance-management responsibilities.

A QRML-based approach brings systematics to using multiple models for trade-off analysis. It provides a modelling framework that is compositional, allows one to identify the relevant aspects, captures management opportunities in a precise way and can be used as a basis for offline and online optimization strategies.

4.2.3.1.4 Application to use case(s)

We have used the QRML tooling to make a high-level model of part of UC4. The model illustrates the modelling concepts provided by QRML and gives an indication of how it can express multi-objective optimization problems that may arise in performance management. Furthermore, we discuss how these models could fit in the TRANSACT architecture and how they could work together with other performance monitoring and prediction models.

The part of UC4 that we consider is the reconstruction of 3D images. From the perspective of a hospital, this task can be performed in the cloud, or on an edge device at the hospital (which is more predictable with respect to timing but has lower quality of the resulting 3D reconstruction). Depending on the current workload (i.e., requests from other hospitals to the cloud), performance constraints, the state of the computational platform, constraints on the operational costs, etcetera, a suitable mode needs to be selected.

QRML has features to capture these aspects and their relations. We have made two QRML models of the Operational Mode Coordinator in the different tiers.

The first model could be a model that is used by the Operational Mode Coordinator in the Edge Tier (see TRANSACT reference architecture). It distinguishes two operational modes: (i) edge-based reconstruction, and (ii) cloud-based reconstruction, each with its own qualities. For instance, the response time for the edge-based reconstruction is better than the response-time for the cloud-based reconstruction, but the quality of the reconstructed 3D image is lower. The numbers for these qualities that are used by QRML for the multi-objective optimization, come from other models, e.g., from simulation-based performance analysis (Section 4.2.2.2) in combination with AI-based workload prediction (Section 4.2.2.1).

Figure 43 shows the textual specification of the model in the DSL. First, several types are declared that are used by the interfaces of the components, e.g., `Reliability`, a Boolean value that captures if the service is delivered reliably or not, and two integer numbers that represent reconstruction quality and response time, respectively, and an integer that represents the requires computation budget. Then, two functions are declared that link into external models; `predict_response_time` determines a response-time prediction from a given computation budget and `predict_edge_load` predicts the available computation budget on the edge node. Then, the component declarations follow. The EdgeNode component, for instance, provides a Computation budget which depends on the predicted value of an external model. The `LocalReconstructor` component requires `Computation` to do the 3D reconstruction. These two components are composed in the `EdgeBasedReconstruction` component. This component also exposes the key qualities of the system. The value for the response time quality is again computed by an external model that indirectly uses the prediction of the load on the edge node. Finally, the main `System` component shows the choice between the two operational modes and models the trade-off between the three qualities. Figure 44 shows a visual representation of the model.

```
typedef Reliability boolean
typedef ReconstructionQuality integer
typedef ResponseTime integer

budget Computation integer

channel Images {
    number: integer
    resolution: integer
}

// External model for prediction of response time
function ResponseTime predict_response_time(Computation)

// External model for prediction of load on edge node
function Computation predict_edge_load()

component EdgeNode {
    provides Computation c
    constraint c = 1000 - predict_edge_load()
}

component LocalReconstructor {
    input Images in
    requires Computation c
}
```

```
component EdgeBasedReconstruction {
    contains LocalReconstructor r
    contains EdgeNode n

    input Images in from r.in
    quality ResponseTime rt
    quality ReconstructionQuality rq
    quality Reliability rel

    constraint r.c runs on n.c

    constraint rt = predict_response_time(n.c)
    constraint rq = 3
    constraint rel = true
}

component CloudBasedReconstruction {
    input Images in

    quality ResponseTime rt
    quality ReconstructionQuality rq
    quality Reliability rel
}

main component System {
    contains EdgeBasedReconstruction er
             or
             CloudBasedReconstruction cr
}
```

Figure 43: DSL specification of example QRML model for the Operational Mode Coordinator in the Edge Tier.

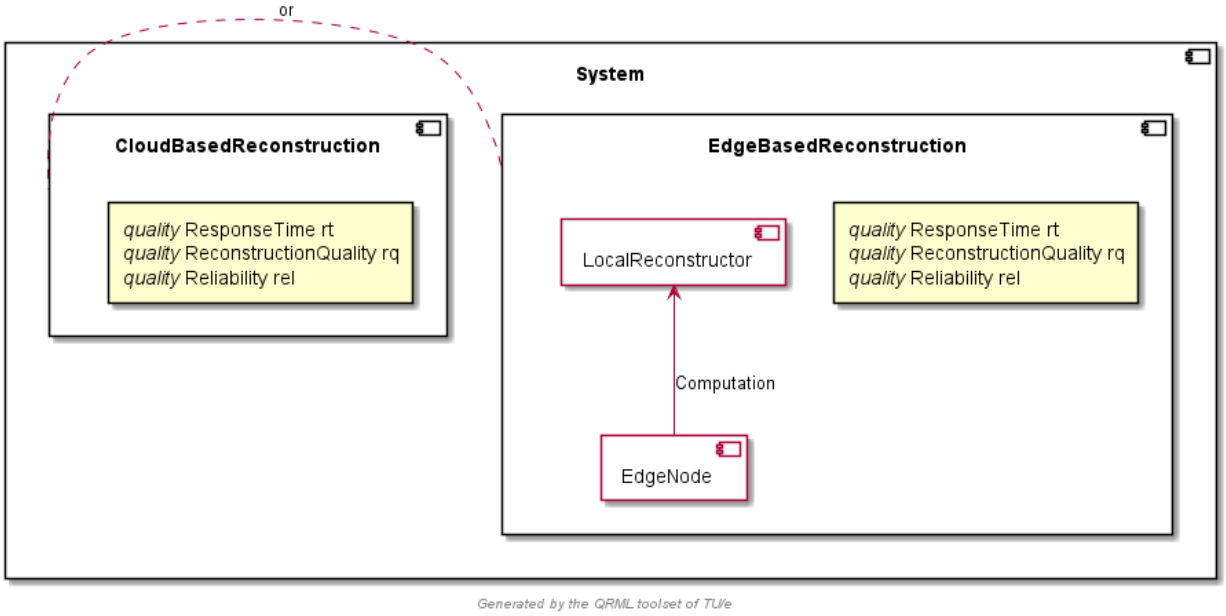


Figure 44: Example QRML model for the Operational Mode Coordinator in the Edge Tier.

The second model is shown in Figure 45 and could be used by the Operational Mode Coordinator in the Cloud Tier (see TRANSACT reference architecture). It also distinguishes two operational modes. The difference between the modes is in the application, and how the application is mapped to the underlying cloud infrastructure (using the Y-chart pattern). The output of this model in the Cloud Tier, is input to the model on the Edge Tier in Figure 44: the `CloudBasedReconstruction` component in Figure 44 is an abstraction of the `System` component as shown in Figure 45. Note that the various mapping options could be manually defined (as is done in this example) or could be generated by other tools (such as the standard Kubernetes Scheduler component). In the latter case, we need to dynamically build the QRML model.

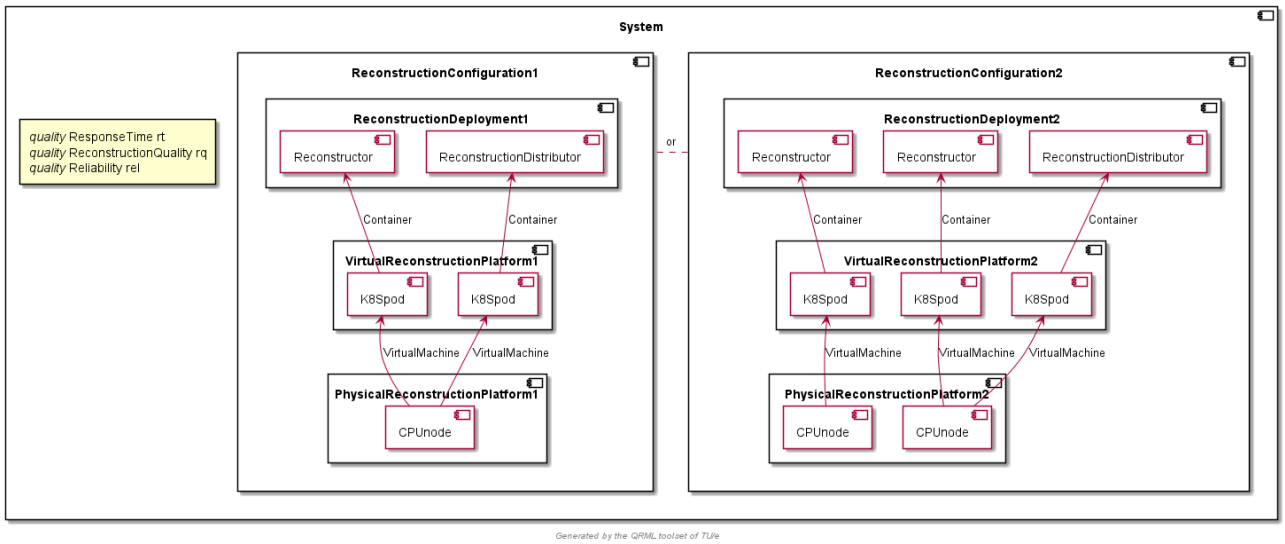


Figure 45: Example QRML model for the Operational Mode Coordinator in the Cloud Tier.

We have identified several challenges that need to be further explored in the project:

- Integration with other models to provide values for the QRML configurations (e.g., predicted response time, quality of 3D reconstruction, operational cost, etcetera).
- Automatic generation and evaluation of (parts of) a QRML model for larger systems (e.g., tens of physical nodes and tens of pods that need to be mapped to these physical nodes; each mapping option can be modelled by a separate choice in the QRML model).
- Heuristics for dealing with large search spaces (for online optimization).

4.3 Solutions for safety and health monitoring, risk analysis, and safety and security assurance

4.3.1 Risk management planning/monitoring

4.3.1.1 Overview

Society increasingly depends on information technologies for achieving its purposes. However, the use of information and communication technologies entails certain risks that must be sensibly managed with security measures that sustain the trust of service users. That is the main reason why companies need to thoroughly analyse the risk involved in their activities, so they can balance risks and encourage opportunities arising from the use of IT.

Being aware of the risks will ensure that systems will work as the Management expects, providing a balanced framework for Governance, Risk Management, and Compliance, three areas that must be integrated and lined up to prevent conflicts, duplication of activities, and no-man's lands.

The tool GConsulting allows consultants and companies to control Complex Management Systems. It monitors and centralises all the information in an optimal way facilitating the location of reports, records, and indicators.

Figure 46 represents the steps that are carried out in order to have a complete risk assessment.

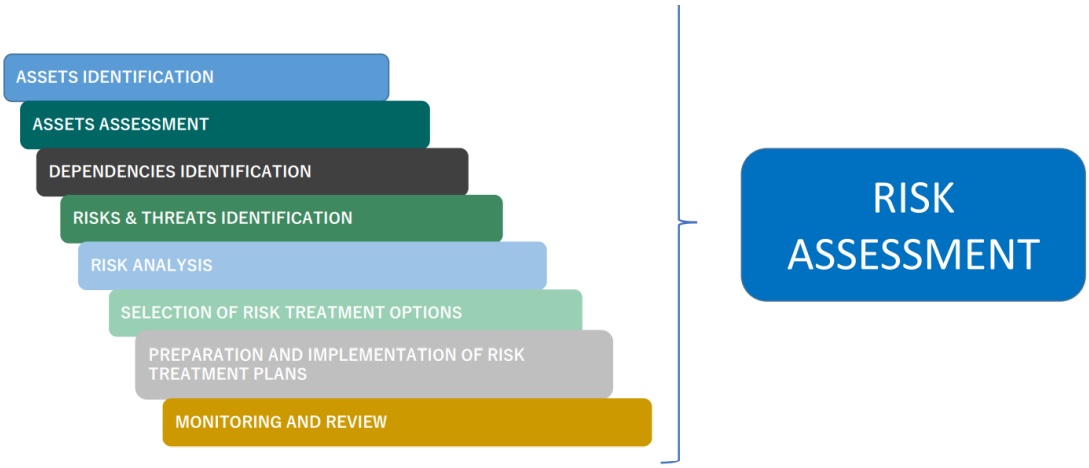


Figure 46: Steps of a Risk Assessment

A complete risk analysis allows companies to protect their mission, as it considers different security dimensions:

- **Availability (D)**: Have access to assets1 when they may be needed [UNE 71504:2008].
- **Integrity (I)**: Assets must have not been modified [ISO/IEC 13335-1:2004].
- **Confidentiality/ Non-disclosure (C):** Asset's information should not be available to unauthorisedly parties [UNE-ISO/IEC 27001:2007].
- **Authenticity (A)**: An entity or asset is what they claim to be [UNE 71504:2008].
- **Accountability (T)**: The activity of the entity or asset can be monitored [UNE 71504:2008].

In TRANSACT project, Singlar is carrying out two different risk assessments in all use cases: (i) The current architecture is being studied in order to assess the risks and potential threats that are applicable, and (ii) There will be a second risk assessment, evaluating the risks and potential threats when the architecture will be in cloud or edge tiers. Finally, Singlar will prepare a report comparing both risk assessments, so the new architecture at least maintains the same risk level as the old one or lower. Risk assessment techniques are an important ingredient of an overall TRANSACT safety concept, which will be reported in detail in the year 2 periodic report.

4.3.1.2 State of the art

There are different methods to manage risks in a company, such as OCTAVE, ISO 27005:2018, or MAGERIT.

OCTAVE

OCTAVE is a risk assessment methodology developed by the Software Engineering Institute of Carnegie Mellon University in the USA. Its methodology is based on three different steps:

1. Build asset-based threat profiles.
2. Identify infrastructure vulnerabilities.
3. Develop security strategies and plans.

ISO 27005:2018 "Information security, cybersecurity and privacy protection: a guidance on managing information security risks"

It formalises the different activities that are needed to take into consideration to enable a company to make the decisions it considers necessary for risk management. The purpose of ISO 27005:2018 is to set some guidelines for managing the risks of information security.

MAGERIT

To develop a thorough risk analysis, MAGERIT methodology can be used, which 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). Thus far, ENISA also accepts other Risk management developed by member states of the European Union, such as in Spain. Over the years, MAGERIT has been developed and improved. It was firstly elaborated in 1997 and CSAE has been working on its

1 We should consider as an asset, everything worth in a company, whether tangible or non-tangible (such as staff, facilities, software, hardware, etc.).

development and updating continuously, taking into account, not only practical experience, but also ISO standards (MAGERIT, 2005)[2].

MAGERIT's purpose is, not only to develop a risk management plan but also to make persons in charge aware of the actual existence of risks and the necessity to manage them using a systematic method, keep the risk low after treating the identified risks, and finally, it prepares the company for certifying processes or audits.

4.3.1.3 Innovation step

Singlar (SNG) have developed GConsulting to lead a complete risk assessment by using MAGERIT methodology. This allows to evaluate if a threat may actually materialise, its probability, and how it could devaluate an asset.

MAGERIT assists consultants and offers some directions about risk management. Companies that apply a risk assessment based on MAGERIT should take the process described on Figure 47 into account:

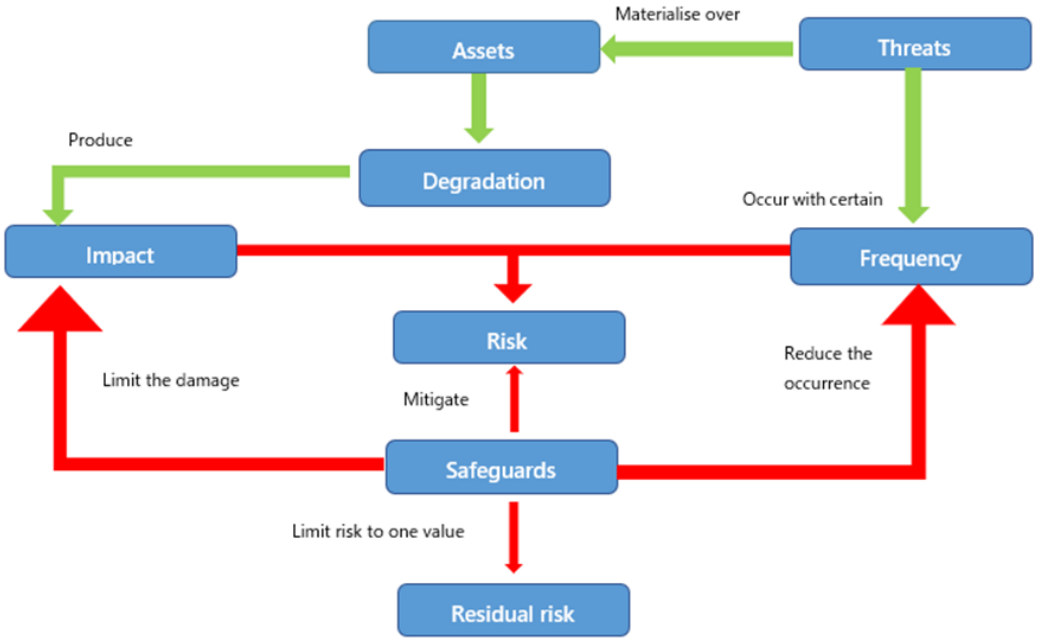


Figure 47: MAGERIT framework

Moreover, Figure 48 shows a screenshot of the tool, where the threats are assessed. After rating the availability (D) and frequency the result is the **accumulated risk** (the accumulated risk is the calculated risk taking into consideration the value of an asset and the value of the assets that depend on it; this value is combined with the degradation caused by a threat and its estimated likelihood of occurrence).

2

Active-threat valuations

Asset	Frequency	D	I	C	A	T
Environment						
LOCALIZATION						
EDAR		100.0	90.0	0.0	0.0	0.0
[I.1] Fire	Rare	Critical	None	None	None	None
[I.2] Water damage	Normal	High	None	None	None	None
[SP02] Electrical installation breakdowns	Despicable	High	None	None	None	None
[SP09] Power outages	Despicable	High	None	None	None	None
[SP10] Theft and vandalism	Despicable	High	High	None	None	None
Equipment						
COMMUNICATIONS						
UBIKITI NANO STATION M2 ACCESS POINT		100.0	0.0	0.0	0.0	0.0
[A.25] Theft of equipment	Rare	Critical	None	None	None	None
[E.1] User Errors	Rare	High	None	None	None	None
[I.8] Failure of communications services	Rare	High	None	None	None	None
UBIKITI NANO STATION M2 ACCESS POINT		100.0	0.0	0.0	0.0	0.0
[A.25] Theft of equipment	Rare	Critical	None	None	None	None
[E.1] User Errors	Normal	Middle	None	None	None	None
[I.8] Failure of communications services	Normal	Middle	None	None	None	None
ATL PLATFORM		100.0	0.0	0.0	0.0	0.0
[A.25] Theft of equipment	Rare	Critical	None	None	None	None
[E.1] User Errors	Normal	Middle	None	None	None	None
[I.8] Failure of communications services	Normal	Middle	None	None	None	None
Ethernet with Modbus-TCP/IP protocol		100.0	0.0	0.0	0.0	0.0
[E.1] User Errors	Rare	Critical	None	None	None	None
[I.8] Failure of communications services	Rare	Middle	None	None	None	None
FIREWALL MERAKI MX65W		100.0	0.0	0.0	0.0	0.0
[A.25] Theft of equipment	Rare	Critical	None	None	None	None
[E.1] User Errors	Normal	Middle	None	None	None	None
[I.8] Failure of communications services	Normal	Middle	None	None	None	None
LoRaWAN NETSGATE COLLECTOR		100.0	0.0	0.0	0.0	0.0
[A.25] Theft of equipment	Rare	Critical	None	None	None	None
[E.1] User Errors	Normal	Middle	None	None	None	None
[I.8] Failure of communications services	Normal	Middle	None	None	None	None
RED PLCs		0.0	0.0	0.0	0.0	0.0
[E.1] User Errors	Normal	None	None	None	None	None
[I.8] Failure of communications services	Rare	None	None	None	None	None
RESPIRÓMETRO ON-LINE SN-8		100.0	0.0	0.0	0.0	0.0
[A.25] Theft of equipment	Rare	Critical	None	None	None	None
[E.1] User Errors	Normal	Middle	None	None	None	None
[I.8] Failure of communications services	Normal	Middle	None	None	None	None
ROUTER TP-LINK		100.0	0.0	0.0	0.0	0.0
[A.25] Theft of equipment	Rare	Critical	None	None	None	None
[E.1] User Errors	Normal	Middle	None	None	None	None
[I.8] Failure of communications services	Normal	Middle	None	None	None	None

Figure 48: Examples of threats and their rating

Nevertheless, GConsulting not only takes into account the threats identified by MAGERIT. It also allows to include certificate-based security solutions. It could monitor and centralise all the information related to the risk assessment. So, this ensures central controlled and monitored access control and function activation to provide a secure diagnostic and function on demand in the companies.

4.3.1.4 Application to use case(s)

The need to implement a risk analysis can be illustrated by the use case "Remote Operations of Autonomous Vehicles for Navigating in Urban Context". This use case is developing a solution that will allow vehicles to be moved from one location to another even without a driver, but a remote operator. As risk analysis allows to know which work elements are subject, it 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; since a risk management process facilitates governing bodies to make decisions considering the risks derived from the use of IT and AI.

Risk analysis has an important role in this use case as there are specific regulations applicable that are already identified in the risk assessment (UN Regulation No. 155 on Cyber Security and Cyber Security Management Systems and UN Regulation No. 156 on Software Updates and Software Updates Management Systems https://unece.org/sites/default/files/2021-03/R155e.pdf).

To carry out the risk assessment, first of all, the use case owner identifies their assets (both tangible and non-tangible, and even their facilities and other locations) and value them according how crucial are for the

service taking into account the abovementioned dimensions. After that, it is necessary to identify which threats can apply to those assets and rates the frequency and how, if occurred, would affect to the different dimensions. Finally, an acceptable risk level will be set, assuming those risks considered as low residual risk. Then, for all the threats that are on a risk level above or equal to the acceptable risk level set, it would be necessary to treat them by applying safeguards, which will be determined by an analysis of cost and benefit (it means, it is necessary that the cost of the safeguards is less than the risk). When the process is finalised on GConsulting, a deliverable is handled: a **Risk Treatment Plan,** where the company can find all the safeguards proposed.

Nevertheless, a risk assessment is being carried on in all five use cases, as it may identify the potential threats involved so an unfortunate event can be avoided, or, at least, the potential damages may be diminished.

4.3.2 Real-time machine-learning based solutions for detecting safety, security, and privacy anomalies

4.3.2.1 Overview

At Singlar (SNG), we use machine-learning technologies to improve the detection of safety, security, and privacy anomalies in industrial applications. We use market leader solutions for Industrial Cybersecurity monitoring, technology Guardian of Nozomi Networks, and we have developed, as part of the project, a middleware called SIRENA to add extra features linking this technology with the Risk Analysis previously explained (included as well in SIRENA platform). SIRENA allows:

1. real-time machine-learning-based monitoring tool with Nozomi (market leader) probe
2. an inventory of assets and alerts in natural language applicable on each use case

Using SIRENA environment with the Risk Analysis embedded, we are developing a platform that relates both the assets and the alerts of the industrial probe, in this case NOZOMI (low level: IP, Mac, ...) with the business information (language of the person in charge of operations or the CISO) that is collected in a CMSS / or the Risk Analysis tool, including the layer architecture to include any other formats in the future such as CMDB Remedy, which is a Configuration Management DataBase (out of the scope in this project).

From there, we can feed the Nozomi probe with metadata (there is bi-directional communication) generating alerts enriched with information related to the machine learning that make the work of technical very agile, to investigate alerts.

Unlike most Computerized Maintenance Management System (CMMS) solutions on the market, our solution incorporates predictive models using the latest machine learning and deep learning techniques in the field of Big Data that must be used for the subsequent processing of these values.

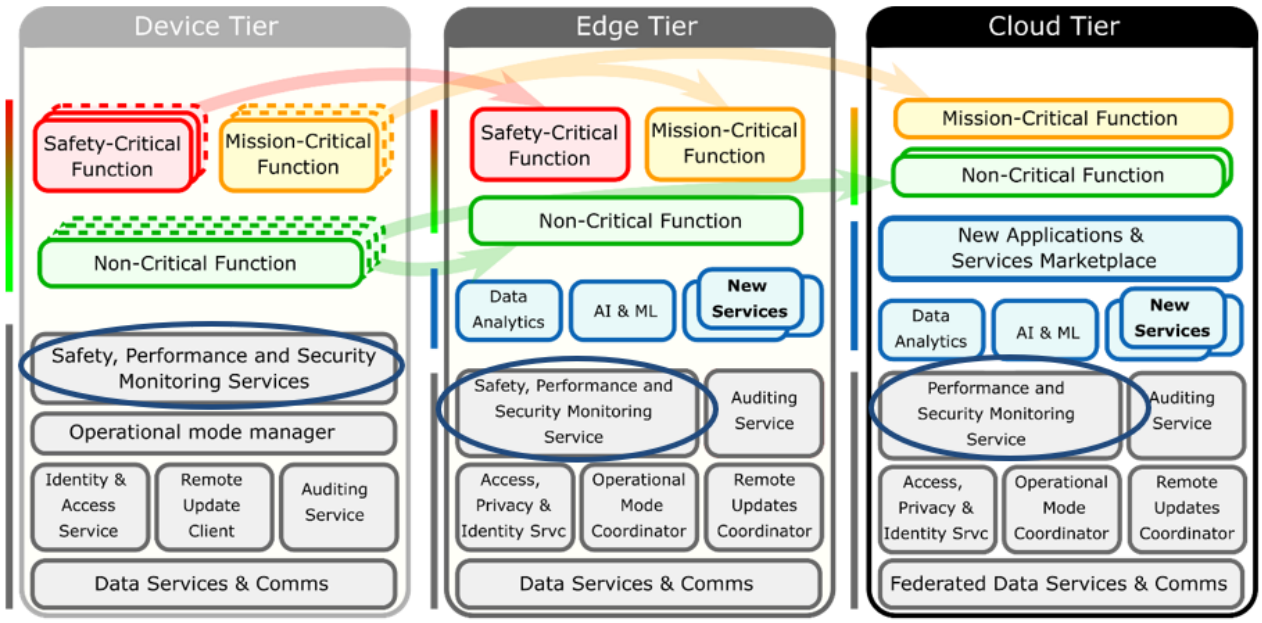


Figure 49: The relation between SIRENA tool and the TRANSACT reference architecture

Our solution uses real-time machine learning and can be used in safety and security monitoring components of the TRANSACT reference architecture as shown in Figure 49. One of the inputs to the tool is a baseline of the desired or normal behaviour. The tool enters a 'threat-detected mode', during which alarms are triggered, based on different patterns:

1. Detection of known attack patterns or undesired behaviour;
2. Detection of anomalous behaviour outside the baseline: Variables out of range, cycles different from normal, change in the orders sent by a SCADA, etc.;
3. Manually programmed alarms for detection of patterns or sensitive values;
4. A library of attack patterns focused on safety and security properties;

4.3.2.2 State of the art

Computerized Maintenance Management Information System (CMMS or CMMIS) is any software that helps management to track all activities and has the capability to identify causes of certain security problems and offers their solutions. However, nowadays, industrial systems must be more predictive and protected against all cyberthreats, masquerade, or misfeasor threats.

We are working on a monitoring system called SIRENA that is nourished from low-level probe that sniffs the real-time network's traffic, with the idea to list/summarise different requirements on security and safety. This technology allows us to monitor the requirements while it also helps and improves security and safety. In short, it:

- identifies the network's components and how they are related.
- identifies the components' IP.
- identifies and, when it is needed, implements a communications protocol.
- starts a 'learning period' so that SIRENA's machine-learning model is trained according to the normal behaviour.
- identifies potential false positives.

- implements an incident (alarm or vulnerability) protocol by sending information about the responsible of the asset defined on the risk analysis minimizing incident response times.

One of the hardest tasks we are currently working on is identifying false positives and classifying a formal list of Operational Technologies (OT), namely, hardware and software that detect or cause a change, through the direct monitoring and/or control of industrial equipment, assets, processes and events, and devices (depending on the type of industry or application).

The real-time monitoring tool detect the attacks (abnormal/undesirable behaviour) by assessing them against the normal behaviour of the network. This tool can also distinguish between legitimate use and a malicious attack. This helps to increase system's protection and avoids/reduces damages to the system and [virtual] properties.

Usually, real-time monitoring tools help companies to comply with a variety of industry cyber-management regulations. Log management is the industry-standard method of auditing activity on an IT network.

There are not many tools specialized in OT environments and each of them are specialized in different industrial environments by detecting different types of devices, but the functionalities are similar in each of them.

The application we have used for TRANSACT project is NOZOMI[3], one of the most popular for its ease of use and for the clear information it shows to the user. It works mainly with assets and nodes, and graphically shows the connections between them.

Another application that is growing its popularity is ARMIS (https://www.armis.com/). A tool that is specially designed to connect with external applications and obtain information about both assets and threats, although the relationship it makes between assets and threats can sometimes be a bit confusing when relating more than one source or destination asset.

Both applications are compatible with SIRENA that is able to connect, obtain the assets and alerts and process them to identify the person responsible, as well as the treatments that must be applied for the threat.

Through the use of rules, SIRENA can also identify assets that NOZOMI or ARMIS have not detected, enriching the original probe information with the business information that SIRENA has received from GConsulting.

4.3.2.3 Innovation step

Figure 50: relation between the tree tools: Nozomi, SIRENA, and G.Consulting.

[3] https://www.nozominetworks.com/products/guardian/

Nozomi Guardian: is a low-level probe to manage an industrial network, with the capability to monitor all devices including ICS, OT, IoT, IT, edge, and cloud assets around the network. This information is filtered and analyzed and then sent to the SIERNA tool to continuously monitor the network.

Up until now, it has been difficult to have comprehensive, real-time visibility into ICS networks, devices, and process status. Without that insight, protecting the control network from cyberattacks and avoiding operational disruptions is a challenge. Nozomi Networks' innovative technology solves this challenge in such a way that is completely non-intrusive and safe for ICS and SCADA networks.

Nozomi automatically discovers the industrial network in real-time, namely, it detects components, connections, and topology of the network from the data that is being transferred over the network. Its advanced learning capability then develops process and security profiles specific to the ICS. Nozomi also uses behavioral analytics to constantly monitor these components. As a result, it can contribute to a rapid detection of cyberattacks and critical process anomalies. Cybersecurity and real-time operational monitoring go hand-in-hand to ensure reliability.

SIRENA: is a tool to provide organizations with extra capabilities for decision making in the event of an anomaly in the operation (either due to cybersecurity or undesired network behaviour). SIRENA has the following features:

1. Semantic Security

Through the use of rules defined within SIRENA, we can identify assets in Nozomi (or in the selected probe) to provide more information, such as applying labels, changing the name of the original asset, including business information received from GeConsulting.

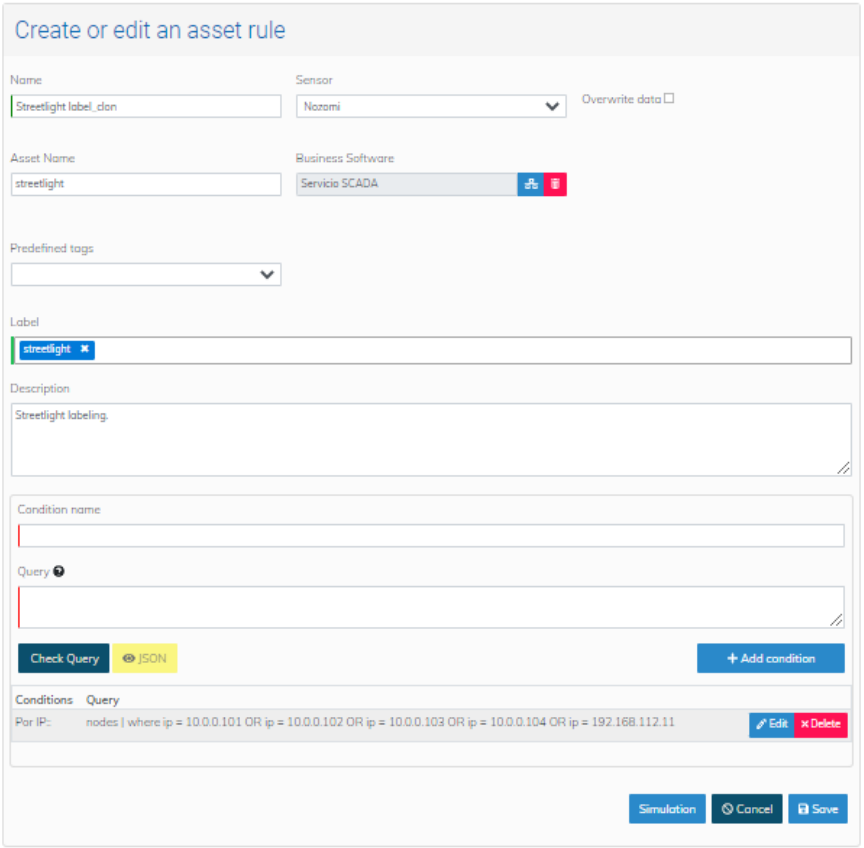


Figure 51: UI for asset definition

As we can see, at the bottom of the page queries are made to Nozomi. These types of queries will be carried out by network specialists who will be able to identify assets that, due to their behavior or the information received by the probes, have not been able to identify the type of asset in question. Once we have a set of results we can rename them, assign them a label and even associate them with an asset name defined at a high level within GConsulting.

2. Business information

Identifying the Nozomi asset with a business name allows us to identify who is responsible for the asset within the company, providing contact information such as phone or email. In addition, it also identifies the threats that have been identified in the risk analysis and therefore what are their treatments when it comes to resolving possible incidents.

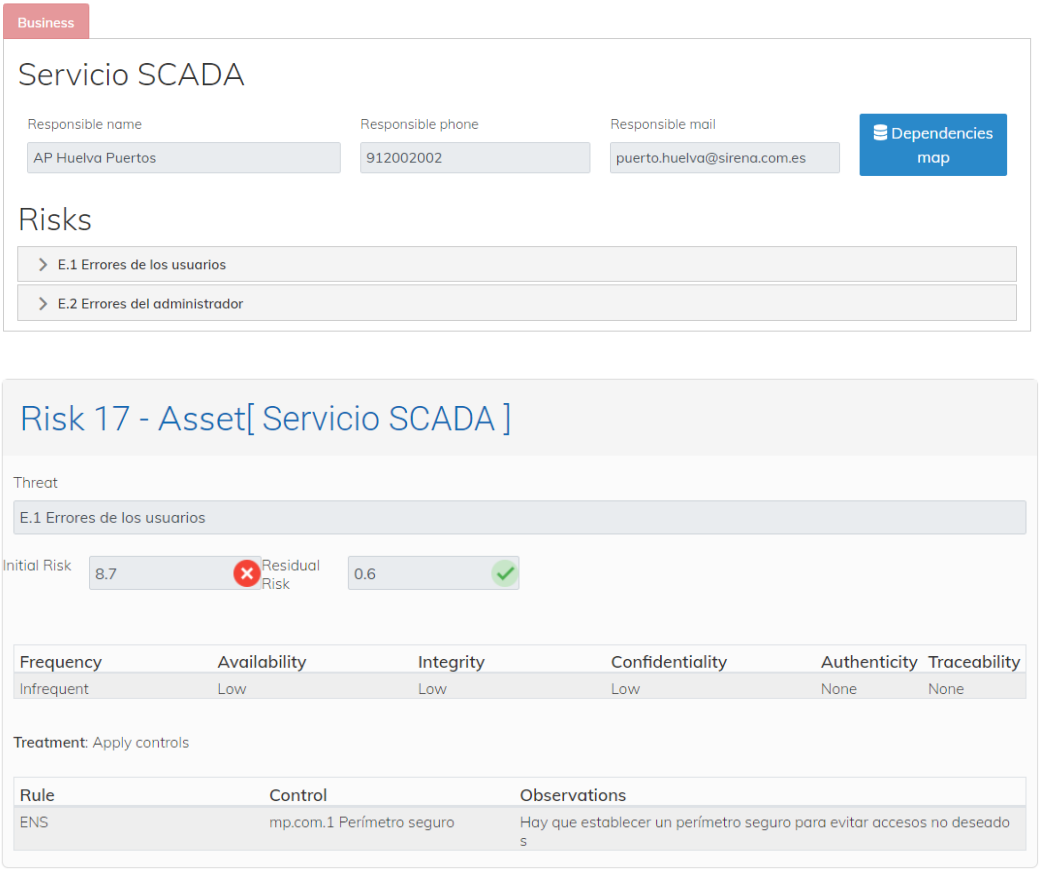


Figure 52: UI for risk definition

3. Interact with industrial cybersecurity probe

All the identification work, enriched with the rules and the business information received from GeConsulting, is returned to Nozomi so that the technicians or SIEMs applications that are going to process that information can reduce the incident resolution time by up to 70% by having the contact information and the treatments that have been defined for the resolution of the incident.

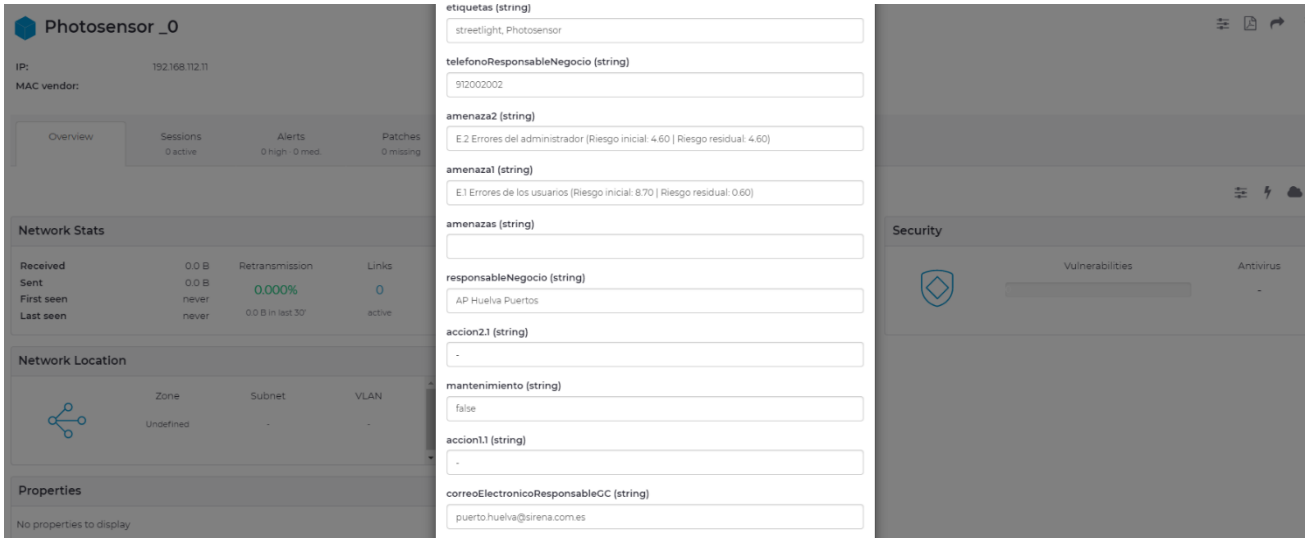


Figure 53: UI for interaction with cybersecurity probe

4. Enrich risk analysis

From the SIRENA alert management, for the assets that are associated with the risk analysis, we can enrich the risks detected by the consultants with the real threats that Nozomi is detecting, making the risk analysis that is a conceptual and subjective vision of the consultant becomes a risk analysis based on real events.

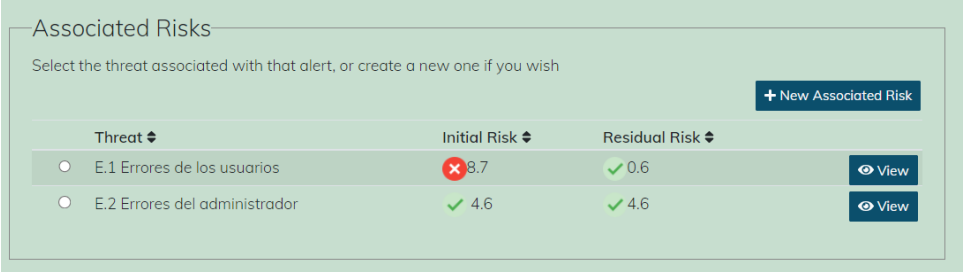


Figure 54: UI for associated risks

With all the Nozomi and GConsulting data, SIRENA manages in a unique interface to monitor and connect, future security issues linked IT & OT devices into specific Tags that represent alerts and assets.

4.3.2.4 Application to use case(s)

One of challenges in TRANSACT project is monitoring safety and security. For example, there could be a malicious attack on an industrial plant in which the attacker takes away the control of the plant processes (e.g., by stopping those processes). When a malicious attack occurs on one or more critical industrial devices (e.g., SCADA) it can have a great impact on both security and safety of the entire system.

From the perspective of safety, an attack (which is caused by security vulnerabilities) could start or stop processes at the wrong time or wrong order, causing damages to properties or persons. Moreover, a security anomaly could also be caused unintentionally or accidentally due to a human mistake.

Next to using security software such as antivirus, guaranteeing the security of a critical network requires using a monitoring tool that, together with a low-level network probing software, is capable of raising alarms and tagging the traffic of the network. Our contribution is to have a real-time monitoring tool to guarantee

an easy answer to security, privacy, and safety threats such as non-authorized accesses, hacked devices, changes on parameters of PLC´s, SCADA, or HMI, and so on.

Therefore, SIRENA solution working together with Nozomi probe and the risk analysis performed at GConsulting are required for all TRANSACT use cases.

For now, risk analyzes have been carried out on all use cases and Nozomi has been installed in UC5, which is receiving information on assets and alerts so that SIRENA can translate low-level information that is difficult for a user to interpret.

4.3.3 Mapping and scheduling techniques across device, edge, and cloud

4.3.3.1 Overview

Edge Computing Platforms (ECP) increasingly integrate applications with mixed-criticality requirements. In our work, we consider that critical applications and Edge applications share an ECP. Critical applications are implemented as periodic hard real-time tasks and messages and have stringent timing and security requirements. Edge applications are implemented as aperiodic tasks and messages and are not critical. We assume that the critical tasks are scheduled using static cyclic scheduling, Time-Sensitive Networking (TSN) is used for dependable communication, and Remote Attestation (RA) is employed to check that the platform components are secure.

We formulate an optimization problem for the joint mapping and scheduling of critical and Edge applications, such that (i) the deadlines of the critical applications are guaranteed at design-time, (ii) the platform has resources to perform RA, and (iii) we can successfully accommodate multiple dynamic responsive Edge applications at runtime. The algorithms are running in the coordinators and managers depicted in the figure below and use the inputs from the monitoring services.

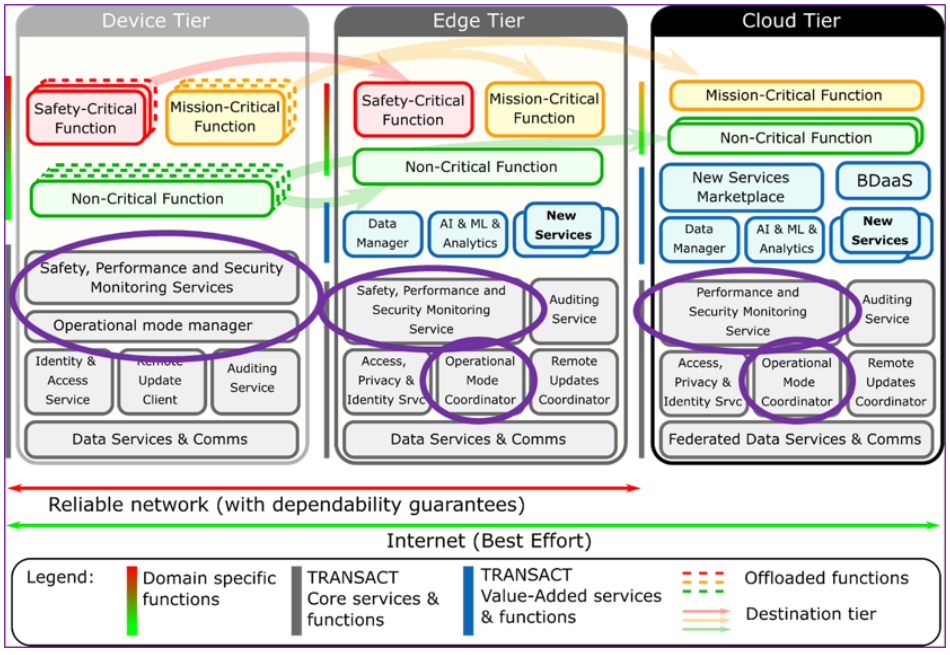


Figure 55: Placement of the resource management algorithms in the architecture

An ECP includes Edge Devices (EDs) capable of communicating and executing computations in the proximity of the “things” and data sources to guarantee effective collaboration between the devices, nodes, and the Cloud. An ED is a compute node that integrates mixed-criticality applications that share the ECP. Regarding

computation, we assume that the critical tasks are running in a Real-Time Operating System using real-time scheduling policies. We consider static-cyclic scheduling in this paper, as it is suitable for applications of high-criticality (Buttazzo G. , 2011). Mixed-criticality applications can be separated in different *partitions* enforced using hardware-supported virtualization, based on hypervisors, such as ACRN or PikeOS, see (Paul Pop, 2021) for a discussion and references. However, this aspect is orthogonal to our problem, i.e., we can consider the partitions during the scheduling and has been addressed in previous work (MOHAMMADREZA BARZEGARAN, 2020).

Regarding communication, we consider that the ECP uses IEEE 802.1 TSN (Group T.-S. N., 2016) as the wired communication solution to connect the EDs, as envisioned by several industrial consortia. TSN, which is becoming the standard for communication in several application areas, e.g., from industrial control to automotive and aerospace, is a set of amendments to the IEEE 802.1 standards, equipping Ethernet with the capabilities to handle real-time mixed-criticality traffic with high bandwidth. TSN supports multiple traffic types, and hence, is suitable for mixed-criticality applications running on an ECP. Researchers and standardization bodies are also working towards extending TSN capabilities over wireless networks (e.g., IEEE 802.11 and 5G). The solution presented in the section can also be extended to consider wireless TSN.

As ECP become more interconnected with the outside world, new attack vectors are possible (T.Pereira, 2017) that may also compromise safety. Therefore, we also consider security aspects in our work. One desirable goal in large heterogeneous ECP's is to assure the integrity of devices. Malicious behavior of devices should be detected in a timely manner and appropriate measures taken, e.g., the filling of a tank should be stopped if the pressure sensor is found to report fake values. One promising direction is to use *Remote Attestation* (RA) to authenticate the hardware and software configuration of a remote device, thus, allowing for the provision of strong assurance guarantees.

RA provides a mechanism to validate the integrity of the software running on untrusted devices. The assumption is that there are both trusted (e.g., high-end EDs) and untrusted devices (e.g., low-end IoT endpoints) in our network. For validation, the trusted party, called verifier, sends an attestation request to an untrusted party called prover. The prover responds with a certificate of its currently running software, e.g., by including a hash of its memory content. Measures are in place such that that certificate cannot be faked by an attacker controlling the prover or the communication link. The verifier checks this certificate, and that the memory content is as expected. If that is not the case, or no response is received to the attestation request, the verifier initiates appropriate measures (out of scope for this work) (Rodrigo Vieira Steiner, 2016).

Researchers have used "fixed priority servers" to integrated periodic and aperiodic applications, which run in the slack of the critical application schedules. Such servers, which are a scheduling concept, are implemented as a periodic task that runs in a core and it is characterized by the tuple $\langle c, t \rangle$ denoting the capacity of the server and the period of the server. Within a schedule that repeats with a hyperperiod H, a server will have several instances which are referred to as server slices. Hence, we consider that Edge applications are scheduled using a special type of server called a *deferrable server*. Deferrable servers use a soft resource reservation technique to allocate its resources to the Edge tasks. Edge application flows are transmitted using the TSN mechanisms specific for strict priority (SP) flows in the windows when ST critical flows are not scheduled. At runtime, the ECP handles Edge applications via monitoring and resource management techniques. Once Edge applications are submitted to run on the ECP, the *controller* ED, which is determined at runtime using mechanisms such as (Vasileios Karagiannis, 2017), receives the submission request. The controller has knowledge on the available resources on the EDs and SWs. It then decides the placement of Edge application tasks on the EDs using resource allocation techniques, e.g., (Olena Skarlat, 2018) which determines the ED that provides the minimum response time for the Edge application.

We formally define the mapping and scheduling problem we address in the paper as follows. Given (1) an ECP modeled with an architecture graph G, (2) a set of critical real-time applications Λ^{crit}, and (3) a set of

verifier/prover pairs, we want to determine a configuration Ψ consisting of: (i) the mapping of tasks to the computational resources in the platform, (ii) the static task schedule tables, (iii) the schedules (Gate Control Lists, or GCLs) for messages, (iv) the period and capacity of the deferrable servers on verifiers, (v) the RA applications Λ^{ra}. Although not part of the optimization problem definition, note that at runtime, our approach handles (vi) the migration of Edge applications to the nodes that have resources for their execution, (vii) the scheduling of Edge tasks on the servers and of their streams on TSN. We are interested in an optimized configuration Ψ such that: the deadlines of all the critical applications are met and the resources available for the RA and Edge applications are maximized, such that we can maximize the RA-based security and minimize the response times for Edge applications.

4.3.3.2 State of the art

Task scheduling in real-time systems is a very well-studied topic, and researchers have considered various scheduling policies, from non-preemptive static-cyclic scheduling, considered in this work, to preemptive dynamic scheduling (Buttazzo G. , 2011). Researchers have also addressed the problem of scheduling traffic in TSN, which requires the synthesis of GCLs (Steiner, 2016). Although initially the work has focused only on messages, recent works propose solutions to the joint task and scheduling problem for TSN-based systems (Niklas Reusch, 2022).

Although there is a lot of work on resource management in Edge and Fog Computing, these do not consider the challenges of mixed-criticality applications that share the same platform. Task and message scheduling has also been considered for Edge and Fog Computing, but typically the focus is on runtime mechanisms for best effort IoT applications, and not on real-time guarantees.

Security in Edge/Fog Computing and IoT has been extensively investigated (Koen Tange, 2020). Remote Attestation is a promising approach to authenticate untrusted remote devices. Remote attestation has been studied in the context of IoT and Fog/Edge Computing, but the timeliness aspects of remote attestation, e.g., that it may lead to deadline misses for real-time applications, have been ignored.

Researchers have investigated the impact of security mechanisms on safety-critical real-time systems, e.g., for resource authentication protocols such as TESLA or for protecting messages with cryptography (Wei Jiang, 2017). However, no approaches exist that evaluate the impact of remote attestation on real-time applications, which are increasingly being implemented using Edge Computing platforms.

4.3.3.3 Innovation step

As discussed in the previous section, researchers have addressed the scheduling of safety-critical real-time applications on ECPs that consider TSN. However, the existing work often looks at computation and communication separately and does not typically consider mixed-criticality applications sharing an ECP. In our work, we are interested in schedules that can integrate both critical and Edge applications, i.e., the deadlines of the critical applications are guaranteed and at the same time, the slack in the schedule (the idle times on processors and links) can successfully accommodate multiple dynamic responsive Edge applications at runtime.

Regarding security, existing research does not consider the impact of RA on the timeliness of real-time applications. Our CP solution synthesizes schedules such that there is enough periodic slack to run regular attestation to secure the ECP. Both TSN and RA place constraints on the scheduling of tasks and messages in the network. RA requires regular slack on verifiers and provers to accommodate the additional tasks and messages and minimize the chance of malware going undetected. Our work considers the impact of RA on the implementation of mixed-criticality applications on Edge Computing Platforms. We have also evaluated our CP approach on a realistic test case related to the horizontal demonstrator.

The ECP is modeled as a directed graph $G = \{N, L\}$, where N is the set of nodes and L is the set of links. A node can either be an ordinary edge device $e_i \in E$, a trusted verifier device $v_i \in V$, an untrusted prover device $p_i \in P$ or a network switch $sw_i \in SW$. The set of links L represents bidirectional fullduplex physical links. Each link $l_{i,j}$ between nodes n_i and n_j is characterized by the tuple $\langle s, d \rangle$ denoting the speed of the link in Mbit/s and the propagation delay in ms. An example ECP architecture graph is shown in Figure 56. The topology is inspired by an industrial use case, in which the network consists of one Edge area, which is connected to production cells containing production lines in a tree-like structure. The devices at the top of the tree are the most powerful in terms of computation power, while the ones at the bottom are least powerful. Verifiers are assumed to be powerful devices from the edge area, while provers are low-end and exposed devices in production cells.

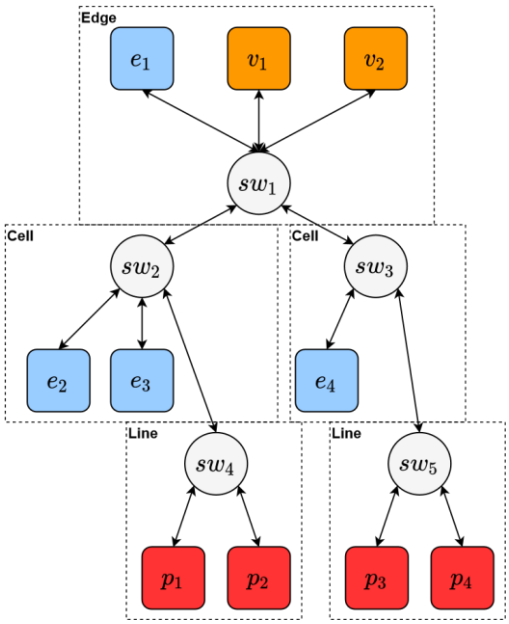


Figure 56: Model of the architecture of the Edge Computing Platform

The fundamental mechanisms that enable deterministic temporal behavior over Ethernet are, on the one hand, the clock synchronization protocol defined in IEEE 802.1ASrev, which provides a common clock reference with bounded deviation for all nodes in the network, and on the other hand, the timed-gate functionality (IEEE 802.1Qbv) enhancing the transmission selection on egress ports.

We detail the Time-Aware Shaper (TAS) mechanism defined in IEEE 802.1Qbv in Figure 57. The TAS is associated with each traffic class queue and positioned before the transmission selection algorithm. A timed gate can be either in an *open (o)* or *closed (C)* state. When the gate is open, traffic from the respected queue is allowed to be transmitted, while a closed gate will not allow the respective queue to be selected for transmission, even if the queue is not empty. The state of the queues is encoded in a local schedule called Gate-Control List (GCL). Our optimization strategy derives these GCLs.

Remote Attestation is performed as follows: A trusted verifier node v_i sends an attestation request to an untrusted prover p_j . In response, p_j invokes some trusted attestation code $AttC$ that measures a region of memory. This measurement is protected using a Message Authentication Code (MAC) with a secret, shared, key K and returned to v_i, which determines whether p_j is in a healthy or compromised state.

The attestation architecture we are using is called SMART (Karim El Defrawy, 2012). It is a hybrid attestation architecture, suitable for low-end low-powered prover devices. In SMART $AttC$ and K are stored in read-only

memory (ROM). The key is guarded by MCU access control rules, such that only $AttC$ can read it. The execution of $AttC$ is non-interruptible and does not leak information. More details about the security assumptions can be found in (Karim El Defrawy, 2012). Furthermore, we assume that SMART is extended with a reliable read-only clock (RROC) as proposed in (Ferdinand Brasser, 2016), to prevent denial of service attacks on the prover.

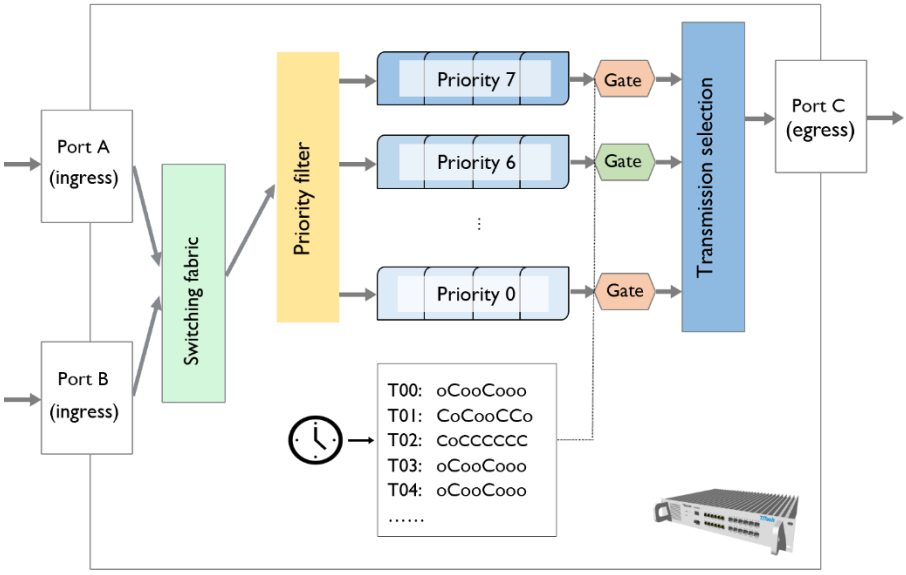
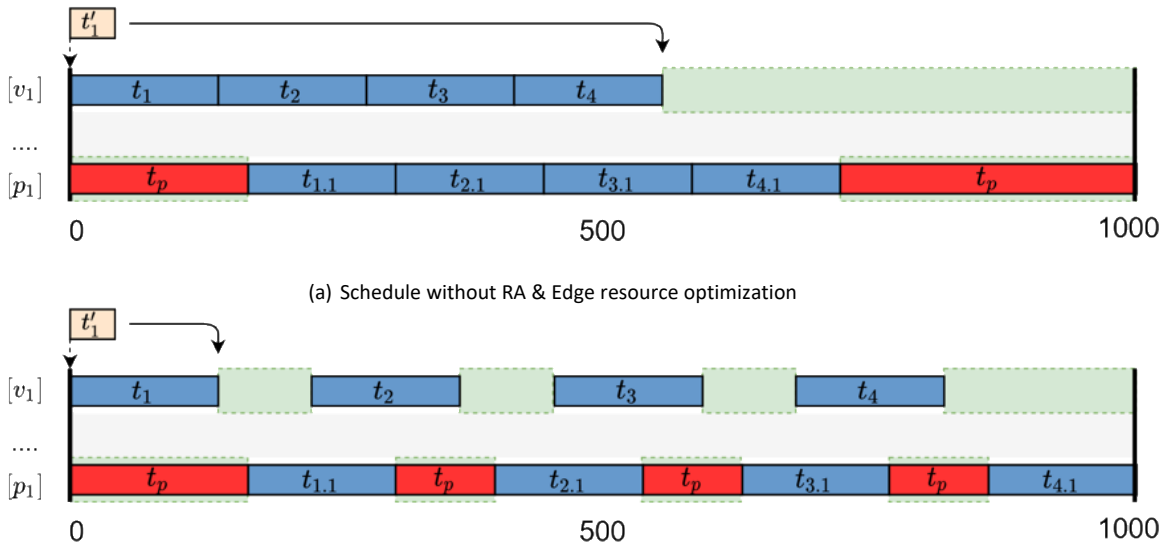


Figure 57: Simplified TSN switch representation

In the basic version of SMART, $AttC$ attests the whole memory at once, in an uninterruptible process. This is infeasible for our platform, since this process would take multiple seconds on low-powered devices, in which no critical real-time task could be executed. Instead, we adopt the ideas from SMARM (Xavier Carpent, 2018). This is a technique that is built on top of SMART. However, instead of attesting the whole memory, the memory is divided into blocks $M_1, \dots, M_n$ of size BS. For a given attestation request, only a certain block M_i is attested, whereby i is determined randomly based on the attestation request. Depending on the frequency and block size, this makes it improbable for malware to hide, given that it cannot predict the memory location that is going to be attested. From the scheduling side, we can choose these parameters appropriately, such that we have a good attestation coverage while also guaranteeing all deadlines of critical tasks. We will choose a block size, such that all remaining slack, after scheduling critical tasks, is used on attestation.

To solve the formulated problem, we use Constraint Programming (CP), which is a method to solve combinatorial optimization problems by expressing them as a set of optimization variables and constraints. CP can find the optimal solution to reasonably large realistic use cases, as we have demonstrated in our previous work focused on the routing and scheduling of secure TSN streams. Due to space reasons, we only detail the constraints on a high level. In our future work, we plan to implement a metaheuristic-based solution to handle larger use cases. Our cost function consists of two equally weighted components: scheduling cost and extensibility cost. The scheduling cost is equal to the sum of all critical application end-to-end latencies. The application latency is defined the distance between the start time of the first task and the end time of the last tasks. We use the “extensibility cost” as a metric to evaluate the capability of a schedule to support RA and Edge applications. Thus, the extensibility prefers task schedules with frequent slack of even size. To achieve this, we maximize the sum of the minimum distances from each task to another. See Figure 58 for an example solution (b) with and without (a) our optimization.



(a) Schedule without RA & Edge resource optimization

(b) Schedule with RA & Edge resource optimization

Figure 58: Example solutions

We minimize the cost function $cost$ (Λ^{crit}) under the following constraints: (1) All deadlines of all applications, tasks & streams are held, and the resource constraints inside the device and edge tiers are satisfied, (2) Tasks do not overlap on their nodes, (3) No frames of any stream overlap on any link, (4) Streams are scheduled consecutively along their route, (5) Frames follow the frame isolation constraint from (Craciunas, 2016), and (6) tasks and streams are scheduled in order, following their dependencies. We use the Cloud tier to run the CP solution and for redundancy of critical tasks (replicas).

4.3.3.4 Application to use case(s)

We evaluate our approach on a realistic use case related to the horizontal demonstrator, but they are also applicable to other use cases. The results show that our approach generates dependable schedules that can meet the timing constraints of the critical applications, have enough periodic slack to perform RA for security, and can accommodate edge applications with a shorter response time.

We created an industrial-inspired test case, part of the horizontal demonstrator. The architecture can be seen in Figure 56. The nodes e_1, v_1 and v_2 are high-powered edge servers running on-premises in a secure room in a factory. v_1 and v_2 were chosen as verifiers as they are the most trusted and protected machines. e_1 is connected to the cloud and responsible for data analytics. The factory consists of two production floors (cells). On the first floor there are two control systems e_2 and e_3 that control a robotic arm p_1 and a conveyor belt p_2. On the second floor there is a control system e_4 that controls an industrial oven p_3 and gets readings from a temperature sensor p_4. The systems p_1 to p_4 were identified as the most safety-critical and should thus be regularly verified. v_1 will do this for p_1 and p_2, while v_2 will do it for p_3 and p_4. Attestation of e_2, e_3 and e_4 could also make sense, but is out of scope for this work, because higher-powered systems can use different RA techniques.

Overall, the use case consists of 19 critical applications, 27 critical tasks and 8 streams. Furthermore, we created 7 sets, labelled E1-E7, of 12 non-critical edge applications, each with 1-2 tasks, half of which exchange non-critical streams. The edge applications implement data analytics and diagnostics and have random arrival times. We ran two experiments: NOEXT without the extensibility cost and EXT with this cost. Our solution was implemented in Python 3.9 using the CP-SAT solver from Google OR-Tools (Inc.).

The results are as follows. The longer a prover stays unattested, the more time an attacker has to execute malicious code and hide itself before the next attestation. In our results, in EXT the average unattested time is reduced by 64.7%, resulting in better security. Additionally, in EXT the average and worst-case response time for dynamic edge application are significantly improved. These improvements come at a slight cost in the form of increased latency for critical applications. However, these applications can still easily fulfill their deadlines. The improvement in response-time for edge applications is also shown by our experiments with the randomly generated set of edge applications E1-E7 in Table 6. For each of these sets we measured the average end-to-end latency both for NOEXT and EXT. On average, the edge applications have 21% smaller latency with our solution.

	NOEXT	EXT
E1	121.75	79.75 (-34.5%)
E2	180.83	107.83 (-40.37%)
E3	215.0	185.17 (-13.87%)
E4	166.25	148.33 (-10.78%)
E5	161.67	134.67 (-16.7%)
E6	109.83	109.0 (-0.76%)
E7	101.5	65.33 (-35.64%)

Table 6: Impact of extensibility formulation on average latency of edge applications

4.3.4 Service continuity monitoring

4.3.4.1 Overview

The TRANSACT project intends to facilitate the distribution of safety-critical applications across device and edge, and for mission-critical applications distribution across device, edge and cloud, while still ensuring end-2-end safety or service continuity. In this section, we focus on *service continuity* for *mission-critical applications* that are (to be) distributed across the device, edge, cloud continuum.

In IT Services, the IT Infrastructure Library (ITIL), is a widely recognized framework of IT service management best practices (Office of Government Commerce (OGC), 2007), IT Service Continuity Management (ITSCM) being one of those. ITSCM aims to manage risks that could seriously impact IT services to ensure that the IT service provider can always provide minimum agreed Service Levels by reducing the risk from disaster events to an acceptable level and planning for the recovery of IT services.

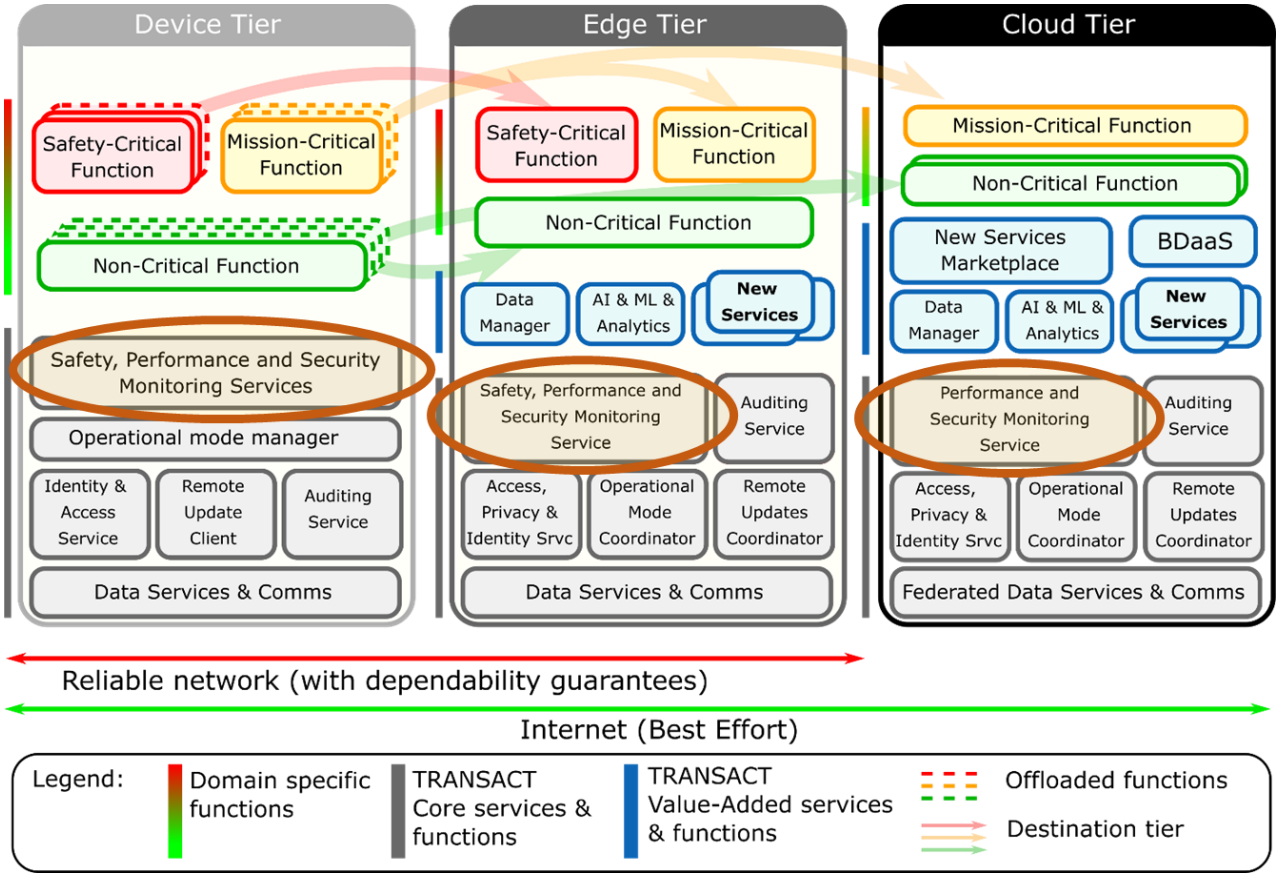


Figure 59: Service continuity monitoring is part of the Safety, Performance and Security Monitoring Services.

When migrating mission-critical applications towards cloud (-assisted) computing, service continuity and service continuity monitoring remain crucial to meet stakeholder expectations and service level agreements in case of disaster / anomaly events. Such service continuity monitoring is part of the Safety, Performance and Security Monitoring Services in the TRANSACT reference architecture (see Figure 59), in particular in relation to mission-critical functions. The services may provide input to the operational mode manager in the device, and operational mode coordinators in edge, and cloud to change into degraded mode operation if necessary.

4.3.4.2 State of the art

Cloud operational monitoring is standard practise at cloud providers (Aceto, Botta, De Donato, & Pescapè, 2013). Nonetheless, minimizing and managing cloud failures, and guaranteeing high levels of availability is still a huge challenge for cloud providers. Unplanned data centre interruptions can occur due to infrastructure or software failures, planning mistakes, human error, or external attacks (Endo, et al., 2017).

Approach	Amazon Web Services	Google Cloud Platform	Microsoft Azure
Monitoring	Amazon CloudWatch permits monitoring resources and services on the AWS cloud. It collects logs and tracks metrics	Google System Health monitors the configuration, activity, and error data to diagnose a potential failure and suggest the best approach for preventing or repairing it.	Azure Monitor, Application Insights, Log Analytics, and System Center Operations Manager handle monitoring.
Distributed storage and redundancy	Amazon Block Storage provides persistent block storage volumes, which are replicated within AWS regions.	Google has multiple points of presence, and Cloud Storage offers multiple availability and latency levels.	Georedundant storage replicates data to a paired datacenter hundreds of miles away
Disaster recovery (DR)	AWS provides IT infrastructure and data DR at the server and storage levels.	Google DR provides backup and recovery of data and applications, as well as DR plan testing.	Azure Site Recovery provides DR for physical and virtual machines with replication and failover.

Table 7: Service continuity management strategies to minimise service failure impact (Endo, et al., 2017).

Table 7 shows the service continuity approaches of three of the large Cloud Providers. Service monitoring is crucial for checking the health of cloud infrastructure applications, and a centre's physical and virtual resources, as part of a Cloud Provider's Service Continuity Management. Data Replication, and Data Recovery strategies also play a key role in these Cloud Provider's approaches.

Multi-cloud environments allow spreading data and applications across a number of separate clouds to ensure that data are secure and loads can be balanced for higher availability (Alshammari, Alwan, Nordin, & Al-Shaikhli, 2017). Still, disruptions could occur, which need to be detected and mitigated at device side. Current cloud systems resiliency research (Prokhorenko & Babar, 2020) focuses on satisfying the growing performance expectations caused by the adoption of Internet of Things (IoT) devices and big data applications. Straightforward expansion of computing power behind cloud services is then not sufficient to cope with the growth of service users and the amount of data required to be processed. For safety-critical systems, the number of connected devices is not that large, however, the safety and mission-critical requirements on service continuity are much stricter.

Summarizing the state-of-the-art, Service Continuity Management and Service Continuity Monitoring has been addressed at cloud-side mostly. Temporary service disruptions may still (infrequently) occur, also due to other factors than the Cloud, e.g., due to internet connectivity or infrastructure disruptions at the premises of the device. However, when connecting safety-critical systems to the cloud, the fail-operational needs for the cyber-physical system's (on premise) usage must be considered.

4.3.4.3 Innovation step

Service continuity management, as part of ITIL, and in cloud services alike, has as goal to make sure services are back up and running within agreed-upon business timelines after major service disruptions. When offloading mission-critical applications to the cloud though, it is not enough to restore services after some

period of time. Also, during those cloud service disruptions, the application itself must be able to continue (albeit perhaps in a degraded-mode, edge-device, or device-only) or alternatively brought to a safe end-state.

In this work, we consider how Service Continuity shall be treated when the solution needs to be fail-operational, even in situations of cloud service disruptions. For mission-critical applications, then the device response to cloud service disruptions must be considered and triggered appropriately. After service restoration, the locally obtained data and results must be synchronised appropriately with the cloud services. For such mission-critical applications, a fail-silent behaviour is most often not sufficient; instead, we strive for a fail-operational solution — see also the concepts for safety-critical platforms in (TRANSACT EU project, 2022) — to continue an application's operation, at minimum to bring it to a safe state.

Integral consideration of service continuity across the device-edge-cloud continuum

The innovation step sought is thus *the integral consideration of service continuity across device-edge-cloud continuum,* in context of a fail-operational solution. Emphasis is on the required distributed service continuity monitoring to enable end-2-end service continuity management. This entails the following three steps:

1. Analyse potential service disruptions and their impact
2. Determine required mitigations and needed monitoring for service continuity across the continuum
3. Design-in a (distributed) service continuity monitoring in a fail-operational solution

For the first two steps, the Systems-Theoretic Process Analysis (STPA) method will be leveraged. STPA (Leveson, 2016) is a hazard analysis technique that 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. 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 them (Leveson, 2016). Tools such as XSTAMPP (Abdulkhaleq & Wagner, 2015) exist for support of STPA analysis. For STPA, service continuity is "just" another level of loss, albeit less critical than safety. Hence, STPA is well-suited for such novel analysis of device-edge-cloud solutions.

Furthermore, it will be considered whether the consideration of service continuity can also be extended towards security continuity. On the analysis side, safety and security analysis are increasingly combined (Lisova, Šljivo, & Čaušević, 2018) (Kavallieratos, Katsikas, & Gkioulos, 2020). STPA concepts apply (Young & Leveson, 2013), and extensions and refinements have been proposed to analyse jointly safety as security risks (Friedberg, McLaughlin, Smith, Laverty, & Sezer, 2017) (Yu, Wagner, & Luo, 2021).

For the third step, we will look at applying a fail-operational concept from the safety-critical systems community to mission-critical device-edge-cloud continuum solutions: the safety-executive concept. The Safety Executive concept (Armoush, 2010) (Douglass, 2005) (Armoush, 2010; Douglass, 2005; Douglass, 2005) targets the problem where a shutdown of the system by the actuation channel itself might be critical or take too much 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 (e.g., a degraded mode of working). The safety Executive Concept provides a centralized and consistent method for monitoring and controlling the execution of complex mitigations in case of failures or service disruptions. The Safety Executive concept has been applied also in systems-of-systems configurations, such as Truck Platooning (Bijlsma & Hendriks, 2017). 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. This concept is also described as an applicable concept for safety-critical platforms in TRANSACT deliverable D3.1 (TRANSACT EU project, 2022).

Figure 60 shows this Safety Executive concept applied to offloading mission-critical functionality to the cloud using the TRANSACT reference architecture (we assume the safety-critical functionality stays at the device, and is for simplicity is omitted from Figure 60). The primary actuation channel of the Safety Executive concept is positioned in the cloud to perform the required functionality. The on-device, fail-safe processing channel is positioned in the device. This channel performs needed fallback functionality in case the primary channel is not available or otherwise not according operating in line with the mission policies. On-device side, the Operational Mode Manager takes on the central role of safety executive coordinating all mission-measures required to switch over to the fail-safe processing channel, or bring the system to a safe state. The service continuity monitoring services check the liveness and proper functioning of the actuation channel and, when compromised, signal an anomaly to the Operational Mode Manager. The Operational Mode Manager controls the Arbiter to signal which of the channels gets used towards the actuators and/or HMI.

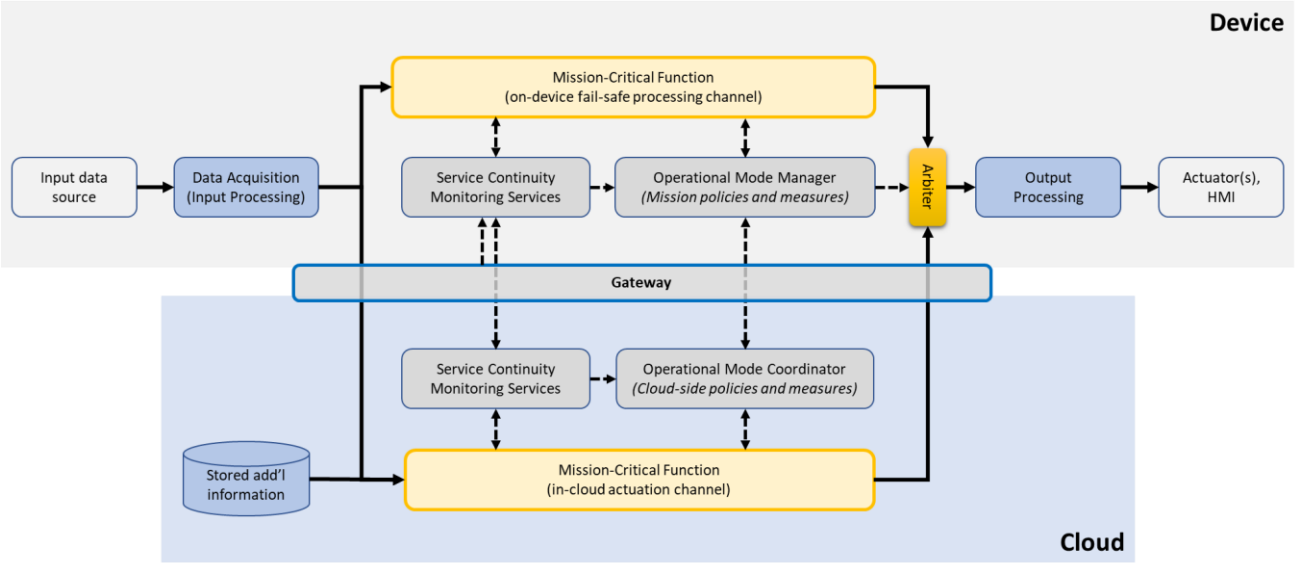


Figure 60: Safety Executive concept applied to offloading mission-critical functionality to the cloud

The on-device fail-safe processing channel is a further channel dedicated to the execution and control of the fail-safe processing. In the presence of a fault in the in-cloud actuation channel, the Operational Mode Manager with help of the in-cloud Coordinator turns off the actuation channel, and instructs the arbiter to have the fail-safe processing channel takes over.

Applying the Safety-Executive concept to device-cloud systems requires on Cloud side also service continuity monitoring and operational mode coordination to be present to manage and shield cloud-side issues where possible from devices. Firstly, many cloud-side availability hick-ups can be managed cloud-side, secondly monitoring of cloud-side functionality needs to instrumentation at cloud-side. These services need to be coordinated and exchange information with the on-device services and Mode Manager in order for the latter to take right actions. The ultimate decision power shall remain on-device though with the Operational Mode Manager.

The solution elaboration in context of use case 4 will consider the necessary monitoring, and service continuity ("safety") polices and measures with required monitoring and watchdog functionality. The key challenges investigated in the solution elaboration for service continuity monitoring are the following:

- How to distribute the mission-relevant operation, redundancy, and monitoring over the device-edge-cloud continuum? Using STPA (Leveson, 2016), the design of system control and monitoring structures can be guided on basis of a conceptual architecture (Levenson, 2019).
- How to effectivity monitor operation in view of grey failures: (cloud) component failures whose manifestations are fairly subtle and thus defy quick and definitive detection (Huang, et al., 2017).
- How to keep service continuity monitoring concepts and solutions stable over edge or cloud-side upgrades meant to support faster innovation and independent releasing of new/upgraded edge or cloud functionality?

Identified monitoring needs could be realised suing similar techniques as reported in section 4.2.1, Performance monitoring. The stability of device-side solutions and implements with respect to cloud-side innovation is a key aspect of the investigation, fostering rapid cloud innovation against stable device-side functionality. Consolidated results of this investigation will be reported in the 2nd and final deliverable of this task T3.3.

4.3.4.4 Application to use case(s)

Monitoring and managing service continuity is essential in all TRANSACT use cases, when mission-critical functionality is offloaded to the edge and/or cloud. When connecting safety-critical systems to the cloud, the an integral consideration of service continuity, including fail-operational needs for the cyber-physical system's usage, is then needed. In the following we illustrate the applicability of service continuity and service continuity monitoring to use cases with an example in context of UC4.

In TRANSACT Use Case 4, adding to medical imaging equipment is augmented with a cloud connection, e.g. for minimally invasive treatment of patients, has the potential to support the task of a surgeon for treatment of patients.

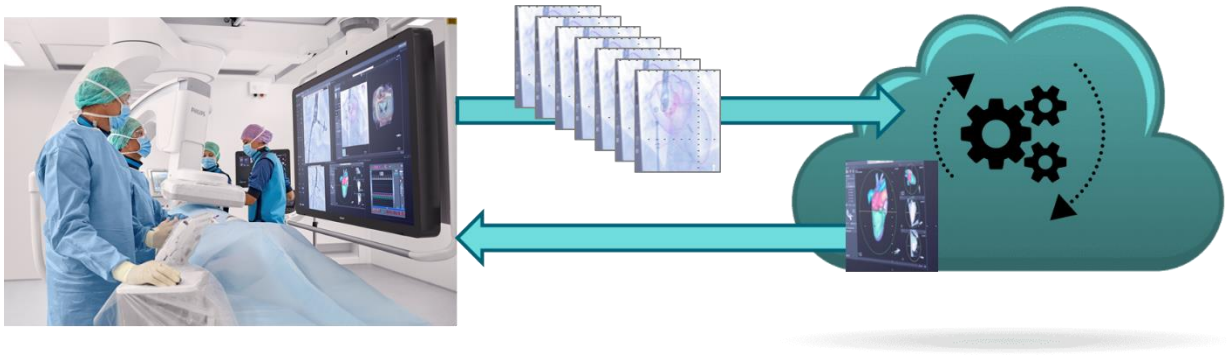


Figure 61: A cloud-connected advanced imaging workflow provides a surgeon access to the latest image processing capabilities

In this use case, the safety-critical applications in such medical imaging (e.g., live X-ray imaging) will remain deployed in the device. However, mission-critical functionality, such as non-real-time image processing, could be deployed advantageously in the cloud, as shown in Figure 61. During treatment of patients, the surgeons may request the cloud to perform the latest innovation, such as an 3D image processing analysis, to provide additional insight for the medical procedure at hand.

Figure 62 shows the example control structure for this scenario. 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 makes a request to an advanced image processing service

from the cloud. At both hospital and cloud side, monitoring services as an operational mode manager and coordinator are in place sync and safeguard the procedure.

Considering service continuity aspects is essential when such complex (cloud-based) image processing comes together with the need for application responsiveness during the live image guided treatment of patients. Also, during a temporary cloud service disruption, the surgeon must always to be able to revert to a well-defined degraded device-only way of working. After service restoration, treatment data and pertinent results again must be made available again to the patient medical records.

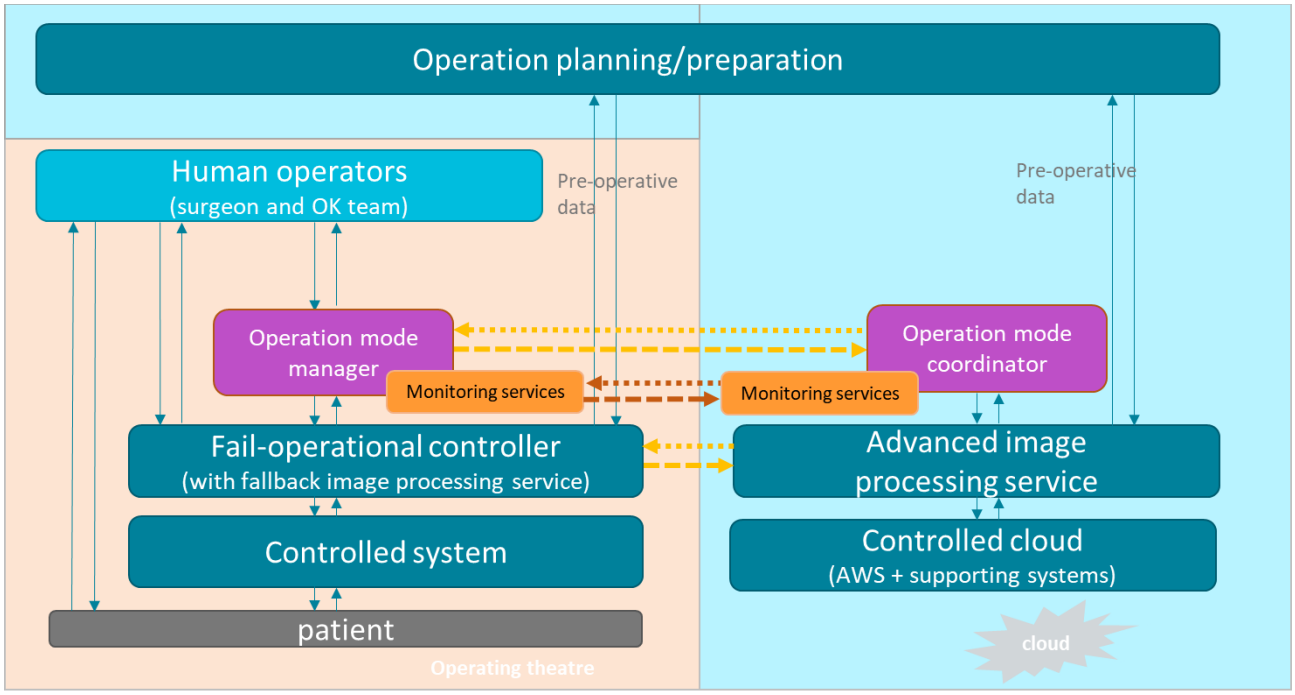


Figure 62: Example model of control structure for the UC 4 cloud-based advanced image processing scenario.

As part of the Use Case 4, service continuity monitoring needs, measures and their distribution over the device-edge-cloud continuum will be elaborated. The impact of service continuity monitoring on the Use Case 4 architecture will be assessed.

5 Platform-related solutions for safety and performance

5.1 TRANSACT project harmonised needs and expectations

Dependable communication is a key component in distributed cyber-physical systems (CPS) which addresses topics such as reliability, timeliness, and availability of information exchange between the system components across all tiers in the device-edge-cloud continuum. To guarantee dependable data transmission between all tiers, partners of the TRANSACT project have proposed solutions for safety-critical platforms (Section 5.2) focusing on the dependability of wireless communications.

Section 5.3, on the other hand, investigates platform-related features and solutions for proactive (as well as reactive) resource scaling on the cloud, ways to achieve higher availability and responsiveness from the cloud providers via better SLAs and special resources such as GPUs.

The selected platform-related solutions are highlighted in Figure 63.

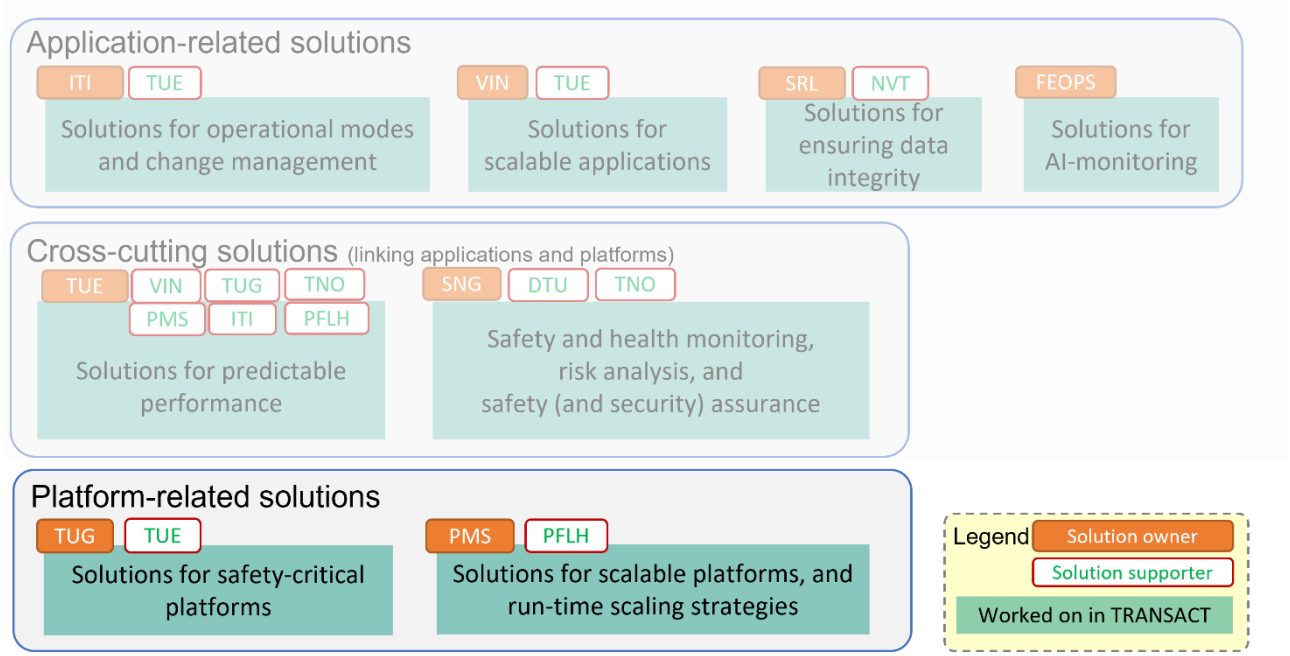


Figure 63: Selected platform-related solutions

5.2 Solutions for safety-critical platforms

5.2.1 Solutions for dependable wireless communication

5.2.1.1 Overview

As described in deliverable D3.1, dependable communication is a key component in distributed cyber-physical systems (CPS). In this context, the term dependability describes the reliability, timeliness, and availability of information exchange between the system components. To meet a system's requirements for safety-critical and mission-critical operation, it is important to guarantee dependable data transmission between all tiers in the device-edge-cloud continuum (see Figure 65 in deliverable D3.1), especially between the device and the edge tier where the safety-critical tasks are performed. We focus on wireless communication as it provides a high level of flexibility in the deployment of the system components

compared to wired connections. Furthermore, there are applications where the use of wired connections is not possible at all, e.g., in applications in which mobility plays an important role.

In distributed systems that rely on data exchange via *low-power wireless* communication, the three criteria reliability, timeliness, and energy efficiency represent a catch-22 dilemma (Schuß, Boano, Weber, & Römer , 2017): increasing the reliability and timeliness of a connection implies less energy-efficiency and vice versa. For example, using retransmissions and higher duty cycle to maximize the communication reliability and timeliness comes at the cost of higher energy consumption. Moreover, wireless communication is exposed to different environmental factors like multipath fading, (cross-technology) interference, and temperature variations, which requires runtime adjustments (e.g., change of physical and link-layer settings based on continuous link quality estimations) to maximize the robustness under dynamically changing conditions (Baccour, et al., 2013). Therefore, we focus on a solution that provides *reliable* communication through effective *interference mitigation*, *timely* communication through *runtime protocol adaptation*, and *efficient* communication through parameterization of *physical layer settings*. In this deliverable, we describe the potential tuning knobs based on state-of-the-art solutions to adjust wireless communication between device, edge, and cloud to sustain given reliability and latency requirements. We specifically focus on low-power wireless protocols like Bluetooth Low Energy (BLE), IEEE 802.15.4 (which specifies, among others, the PHY and MAC layer of ZigBee), as well as the new IEEE 802.15.4z standard (which defines the PHY of low-power devices communicating using ultra-wideband technology). We will focus on solutions based on the *monitoring and adaptation* concept as described in deliverable D3.1, but also consider the option of combining it with solutions based on *time-triggered* concepts or *synchronous transmissions*. Therefore, we will refine our solution in the second version of this deliverable (D3.5).

5.2.1.2 State of the art

Solutions for dependable communication that rely on the monitoring and adaptation concept, also referred to as adaptive communication protocols, typically rely on monitoring performance metrics (e.g., latency) or link quality estimators (e.g., packet delivery ratio) to adapt communication parameters accordingly. Classical metrics that are used to capture the characteristics of wireless links are, for example, the packet reception ratio (PRR), the received signal strength indicator (RSSI), and signal-to-noise-ratio (SNR) (Baccour, et al., 2012). The IEEE 802.15 standard also foresees a link quality indicator (LQI), but its implementation is vendor-specific. Besides the impact of multipath fading and temperature, external radio interference is a major problem for wireless communication. For protocols operating in the license-free 2.4 GHz ISM band, such as BLE and ZigBee, cross-technology interference caused by IEEE 802.11 (Wi-Fi) traffic is most critical because of its much higher transmission power and channel bandwidth. Thus, heavy Wi-Fi traffic negatively affects the communication performance (i.e., the latency, throughput, and energy consumption) of protocols using a lower transmission power and bandwidth (Baccour, et al., 2013). Furthermore, Wi-Fi interference can also be problematic for protocols operating in the 5 GHz and 6 GHz band, such as UWB. (Bellorado, Ghassemzadeh, Greenstein, & Sveinsson, 2003) and (Firoozbakhsh, Pratt, & Jayant, 2003) use simulations to evaluate coexistence issues between IEEE 802.11a and UWB in the 5 GHz band, and their results highlight the need for interference mitigation techniques in UWB systems. Such coexistence problems between UWB and Wi-Fi are also expected in commercial low-power UWB devices operating in the 6 GHz band, due to the release of the of the new Wi-Fi 6E standard (an extension to IEEE 802.11ax that allows Wi-Fi to operate at 6 GHz) (Swedberg, 2020).

As interference is highly dynamic and continuously changes over time, it is important to consider this factor in the design of an adaptive protocol. To perform interference detection and mitigation, constrained devices need to be able to acquire additional information in a simple and energy-efficient way. Furthermore, the applied interference models need to be lightweight such that they can be parameterized at runtime. A common way to measure and determine the type of interference in a frequency channel is to perform noise

floor measurements (Baccour, et al., 2013). The information from these measurements is then used to derive interference models. An example for a quite simple and lightweight model is the two-state semi-Markov model, in which the current state of a channel is classified as in-use (busy) or free (idle). The parameterization of this model at runtime can be used for interference estimation and runtime adaption of protocol parameters. (Brown, Roedig, Boano, & Römer, 2014), for example, analysed the distribution of the measured idle times to perform PRR predictions, and (Noda, Prabh, Alves, Boano, & Voigt, 2011) used such an RSSI sampling technique to derive a new channel quality metric that could be used for interference aware protocols in wireless sensor networks (WSN). In general, common interference mitigation techniques can be divided into the following categories: frequency diversity, space diversity, hardware diversity, redundancy, and time diversity. The most common techniques rely on frequency diversity, e.g., BLE uses adaptive frequency hopping (AFH) per default to avoid continuous interference in a single channel. A more detailed description on interference measurements, modelling, and mitigation can be found in (Baccour, et al., 2013).

As previously mentioned, runtime monitoring and adaption is important as the quality of wireless communication depends on frequently-changing environmental factors. Communication standards already foresee different mechanisms that allow to adjust the performance depending on an application's needs. BLE, for example, supports different physical layer modes as well as dynamic changing of the connection parameters, which allows to adapt the throughput, reliability, and energy consumption based on the application requirements (Spörk, Classen, Boano, & Römer, 2020). However, the information on how to turn these tuning knobs depending on the application at hand is often missing, and also the potential of runtime adaptations is often not exploited. State-of-the-art research focuses on developing and evaluating solutions to this problem by finding the best combination of monitoring certain metrics and adjusting different protocol parameters to achieve a certain goal. We will provide a brief overview on solutions related to BLE and the IEEE 802.15.4(a/z) standard, as they are omnipresent technologies in today's Internet-of-Things (IoT) landscape.

BLE-related solutions: (Spörk, Classen, Boano, & Römer, 2020) provide a solution based on monitoring the BLE link quality and dynamically adapting the BLE physical layer (PHY) at runtime to improve communication reliability while keeping the energy consumption low. They also present solutions relying on blocking or unblocking channels (channel management) that can be used for the AFH. (Pang, et al., 2021) also worked on a channel management solution to increase the robustness of BLE connections, but, in addition to link quality monitoring, they also introduced a novel interference awareness scheme as well as an improved channel selection algorithm (CSA); the CSA is used by the AFH mechanism to determine which channel to use next. The channel management solution proposed by (Poirot & Landsiedel, 2021) does not focus on changing the CSA, but on the way channels for the AFH are included and excluded. (Park, Lee, & Bahk, 2019) use a different approach compared to the channel management and provide an algorithm for data rate and transmission power adaptations to achieve given reliability requirements while minimizing the energy consumption. In (Spörk, Boano, & Römer, 2019), the goal was to improve the timeliness of BLE communication by adapting at runtime BLE connection parameters such as the connection interval and slave latency. The authors identify the impact of interference by the number of connection events needed to successfully transmit data in a standard-compliant fashion (Spörk, Schuß, Boano, & Römer, 2021)[OBJ], where the authors use BLE connection parameter adaptation for meeting end-to-end dependability requirements, such as end-to-end latency and end-to-end reliability, between device and cloud.

IEEE 802.15.4 (a/z)-related solutions: (Lin, et al., 2006) present an algorithm to adapt the transmission power in wireless sensor networks according to changing environmental conditions, which helps to save energy and improve communication robustness. (Boano, Römer, & Tsiftes, 2014) and (Sha, Hackmann, & Lu, 2013) provide solutions for runtime adaptations of the clear channel assessment (CCA) threshold. The former focuses on the impact of temperature variations on communication robustness, and the latter on the operation in noisy environments. (Zimmerling, Ferrari, Mottola, Voigt, & Thiele, 2012) demonstrate a runtime

parameter adaptation for low-power medium access control (MAC) protocols that can automatically translate application-level requirements (e.g., reliability, latency, and network lifetime) to MAC parameters, which replaces the manual MAC configuration in case of network changes. (Park, Fischione, & Johansson, 2010) also show an adaptive MAC algorithm that considers the total power consumption, reliability, and latency requirements, and adjusts the MAC parameters (i.e., minimum value of the backoff exponent, maximum number of backoffs, and maximum number of retries) accordingly. To increase the dependability of UWB Systems, (Großwindhager, Boano, Rath, & Römer, 2018) designed a runtime adaptation scheme for UWB PHY settings that is based on channel impulse response measurements for accurate link quality estimations.

Although research already provides good solutions for adaptive wireless protocols that can guarantee reliable, timely, and energy-efficient data transmission, their application in real systems is not yet common today. Parameters are still often hardcoded and no runtime adaptations performed. We want to change this in the TRANSACT context. Furthermore, in distributed CPS based on a device-edge-cloud architecture, like the TRANSACT reference architecture, also the end-to-end communication requirements between device and cloud should be considered. However, most of the described solutions only consider the local network part, i.e., device to edge, but not the end-to-end requirements between device and cloud. Therefore, we want to integrate runtime adaptation of low-power wireless networks into the TRANSACT context, while considering the dependability for the whole device-edge-cloud continuum.

Besides that, we also investigate how interference caused by the new Wi-Fi 6E standard will affect ultra-wideband (UWB communication and ranging performance, as UWB is an emerging technology for IoT applications that require accurate position information (Alarifi, et al., 2017). Typical use cases for UWB are, e.g., hands-free access control, real-time location services, object tracking, and indoor navigation (Pirc, 2021). As such location-based services are often used in safety-critical domains (e.g., production floors), it is becoming increasingly important to strengthen the dependability and security of UWB technology (Stocker, Großwindhager, Boano, & Römer, 2020).

5.2.1.3 Innovation step

The heterogeneity of technologies used for the data transmission between device, edge, and cloud makes the maintenance of dependable communication quite 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. As described in Section 5.2.1.2, there are different ways to design solutions for adaptive communication protocols depending on the selected protocol. However, each of these methods comes with different trade-offs, e.g., in terms of energy and memory efficiency as well as computational and communication overhead. We will evaluate how monitoring different connection metrics in the device-edge-cloud continuum and adapting various communication parameters according to the gathered information can improve the timeliness, reliability, and energy-efficiency of communication in the TRANSACT context.

As the proposed solution strongly depends on the selected protocols, we currently consider a BLE connection between device and edge, and a 4G (LTE) connection between edge and cloud. The selection of these technologies is based on the planned integration of this solution in UC3 (see Section 5.2.1.4). We will take the work of (Spörk, Schuß, Boano, & Römer, 2021) as a baseline, as it is among the first providing a solution that considers end-to-end dependability requirements between device and cloud, and extend their work to fit into TRANSACT. We plan to investigate the potential of different physical and link-layer adaptations, such as adjusting the transmission power or payload length of packets on a per-channel basis, to meet the requirements for safety-critical operation. Based on the results of our evaluation, we will refine our solution until M33 in deliverable D3.5.

Besides tackling the issue of dependable communication for protocols operating in the 2.4 GHz band, we have already performed an experimental evaluation about the impact of Wi-Fi 6E interference on UWB communication and ranging performance in the 6 GHz band that was published at IPSN'22 (Brunner, et al., 2022). Compared to existing literature, our work does not rely on simulations and the experiments are performed under real-world conditions in a large-scale indoor testbed on off-the-shelf UWB hardware (DW1000). The results are the first to practically show that Wi-Fi 6E interference can cause up to 100% packet loss and that the chances for successful two-way ranging using UWB devices get as low as 4%. To mitigate the impact of Wi-Fi 6E interference on UWB systems, we have proposed the following countermeasures: an optimal selection of physical layer settings as well as the use of a tight synchronization to prevent false detection of Wi-Fi 6E traffic as UWB fames and an overshooting of the radio's automatic gain control. We further devised a technique to detect the presence of Wi-Fi 6E interference directly on UWB hardware and defer UWB transmissions accordingly, as clear channel assessment is not supported on UWB radios. Our experimental evaluation shows that the proposed solutions provide effective interference mitigation, allowing to increase the reliability of UWB systems by up to 47% in harsh RF settings.

5.2.1.4 Application to use case(s)

In general, every use case that uses low-power wireless communication and has requirements on the reliability and timeliness of the communication between different tiers could benefit from the solutions presented in Section 5.2.1.2. To demonstrate that our specific solution fits into TRANSACT, we will integrate it in Use Case 3, which focuses on the implementation of a cloud-featured battery management system (BMS). Although today's in-vehicle communication is still based on wired connections such as the CAN bus, the flexibility of wireless connections is attracting the attention of the automotive industry. As shown by (Rincon Vija, Cregut, Papadopoulos, & Montavont, 2021), for example, the applicability of wireless connections for BMS in electric vehicles is definitely a promising option and will become increasingly popular.

We demonstrate our solution by sending data via BLE from a device (e.g., the battery itself or a sensor node) to the in-vehicle gateway (edge) that forwards the converted data via LTE to the cloud, while maintaining given dependability requirements. To also provide a full experimental evaluation of the functionality and potential of our solution, we will perform tests in our D-Cube[4] based testbed at TUG (Schuß, Boano, Weber, & Römer, 2017).

Furthermore, we have already successfully evaluated the potential of our Wi-Fi 6E interference mitigation technique for UWB systems operating in the 6 GHz band in a large-scale testbed at TUG. The details about the experimental setup, the performed evaluation, and the proposed solutions can be found in our paper (Brunner, et al., 2022).

5.3 Solutions for scalable platforms, and run-time scaling strategies

5.3.1 Overview

In image-guided therapy, more and more advanced and compute intensive algorithms are needed to provide optimal patient care. The typical solution to solve this via computation resources in the intervention room is very costly because:

[4] https://iti-testbed.tugraz.at/

- Expensive computation resources are needed to meet the needs for quick results. Typical response times needed are, depending on the functionality, within a few seconds to a few minutes. For this reason, GPUs are commonly used too.
- The low usage of such resources (<5% of the time)

Availability of algorithms needs to be extremely high. Patient treatment happens during an examination in an image guided therapy room, and the results must be available immediately.

Especially the sparse usage pattern of the compute resources in image-guided therapy raises a need for a more distributed solution that allows for sharing resources over multiple intervention rooms (and even hospitals). Note that in many cases, there are a small number of intervention labs in one facility (e.g., three or fewer). In such cases, sharing within facility is still not cost effective. A complicating factor is that the amount of data can be quite heavy, up to 500 Mbps at peak loads.

This all results in the following list of high-level requirements:

- Need for pro-active scaling with GPU support to meet response times. It is too late to start up computation instance when request for computation comes in (reactive scaling).
- Flexible scaling solution for low and high loads. During non-office hours the load will be minimal, and even during office hours load can be quite limited at times.
- High availability, 99.9%. This means a reliable scaling algorithm with typically some over-scaling to meet this. Also, network QoS management is part of the complete solution.
- Pro-active monitoring of the solution, to be able to solve potential issues without patient treatment impact.

5.3.2 State of the art in typical scaling solutions

5.3.2.1 Orchestrator

Typical scaling solutions all require some kind of orchestrator that orchestrates the scaling process of workers, while the workers fulfil the active requests.

When we look at state of the art, then the workload is usually encapsulated in containers. So, the orchestrator would be a container orchestration system. This is where Kubernetes comes in as state-of-the art portable and extensible open-source platform for managing containerized workloads and services.

The following topics describe what is available within Kubernetes regards to scaling.

5.3.2.2 Horizontal pod scaling

In Kubernetes, Pods are the smallest deployable units of computing that you can create and manage. A Pod is a group of one or more containers with shared storage and network resources. A Pod models an application-specific ‘logical host’.

Horizontal scaling means that the response to increase load is to deploy more Pods. This is different from vertical scaling, which would mean assigning more resources (Memory/CPU) to the Pods that are already running for the workload.

5.3.2.3 Reactive scheduling

In Kubernetes, reactive scheduling is done using the Horizontal Pod Autoscaler (HPA). The HPA periodically queries the running Pods for metrics marked for scaling (usually resource metrics like CPU) and averages the values. Each Pod specification (usually in the form of a Kubernetes Deployment) also has the desired values for these metrics.

The number of replicas (how many of the same Pods) to run is than calculated as:

*DesiredReplicas = ceil(CurrentReplicas * (CurrentMetricValue / DesiredMetricValue))*

This is the most basic implementation, and there are many more conditions taken into account, however that would include too many details here of Kubernetes.

Summarizing, the HPA controls periodically the number of replicas based on defined metrics, which are usually resource metrics like CPU or number of requests (web app).

5.3.2.4 Node scaling

When we talk about Pod scaling, we ignored the fact that we need worker nodes to run our Pods on. This is what's called node scaling.

Each node comes with its own resources (CPU/Memory) and that limits the number of Pods that can be scheduled on this node.

When we would install Kubernetes in an on-premise server, node scaling is usually not applicable. It's often not possible to automatically add new worker nodes to our cluster. However this is something that cloud enables us.

In the cloud we can request a new compute instance and add this as a worker node to our cluster.

Karpenter is a Kubernetes operator that automatically launches just the right compute resources to handle your clusters applications. Karpenter is an example of how node scaling is implemented for the AWS cloud. For other cloud providers similar solutions exists.

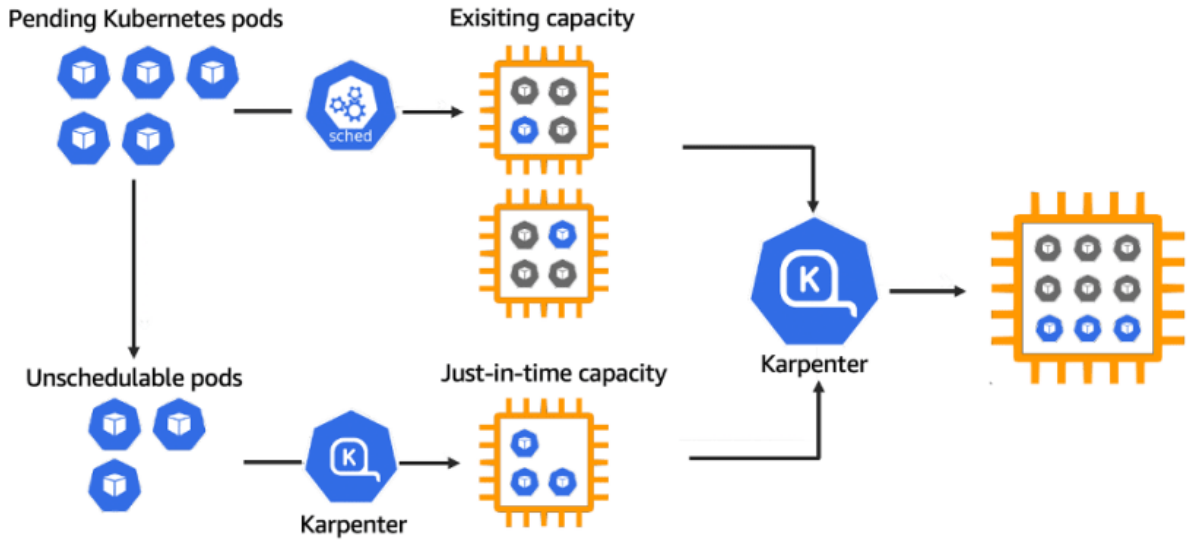


Figure 64: Karpenter and Kubernetes

This node scaling uses the concept that you can request Pods that currently do not fit your existing capacity. These Pods will become unschedulable and this is where new node scaler will act upon.

5.3.2.5 Specific resources

All the previous sections mention compute resources. This in general means common compute resources like CPU and memory. These resources are so generic that all scaling solutions have incorporates smart ways to deal with these resources. Both memory and CPU time both have a relative high resolution to assign to

workloads, e.g., Memory in bytes, CPU in milicores (1/1000th of a CPU core), and so on. This allows for fine grain resource specifications and scheduling.

If the computation requires more specific resources (e.g., graphics processing unit (GPU)) then the scheduler must take these requirements into account as well. The challenge with these specific resources is that they usually have a resolution of 1 (e.g., 1 GPU) and don't allow for fractional usage. This makes scheduling more constrained and less flexible to handle various loads.

When we look at HPA, having a unit of 1 will not help us much as we requested a single GPU and are using a single GPU, this ratio will always be 1.

There are multiple reasons why the unit for these specific resources usually is one, some reasons (but not limited to):

- Combined resource (e.g., GPU consist out of compute and memory)
- Dedicated interface (e.g., application needs to talk to a specific hardware interface with no virtualization layer in between)

For special use cases solutions exist to do fraction GPUs. However, these are usually tailored to sharing a single GPU for multiple virtual machines that require video rendering. Not for generic computing on the GPU.

5.3.3 Innovation step

To overcome the challenges as described in the overview we need to look at a combination of components.

5.3.3.1 KEDA vs HPA

As indicated in previous sections, the Horizontal Pod Autoscaler scales based on loads, so reactive to the requests. This does not meet the requirements on response times, as it takes significant amount of time to start new instances (typically more than 1 minute). So, we need pro-active scaling, scale our compute up before the load arrives. Fortunately, this is possible since the state of the procedure in the intervention room is known, this knowledge can be used for scaling purposes.

This is where Kubernetes Event-driven Autoscaling (KEDA) is designed for. KEDA supports events from different sources (currently 56 sources are supported) to horizontally scale your pods. KEDA will monitor the service and based on the events that occur it will automatically scale your resource out/in accordingly.

Behind the scenes, KEDA acts to monitor the event source and feed that data to Kubernetes and the HPA to drive rapid scale of a resource. Each replica of a resource is actively pulling items from the event source.

For example, if you wanted to use KEDA with message queue as event source, the flow of information could be:

- When no messages are pending processing, KEDA can scale the deployment to zero.
- When a message arrives, KEDA detects this event and activates the deployment.
- When the deployment starts, one of the containers connects to the message queue and starts pulling messages.
- As more messages arrive in the queue, KEDA can feed this data to the HPA to drive scale out.
- Each replica of the deployment is actively processing messages. Very likely, each replica is processing a batch of messages in a distributed manner.

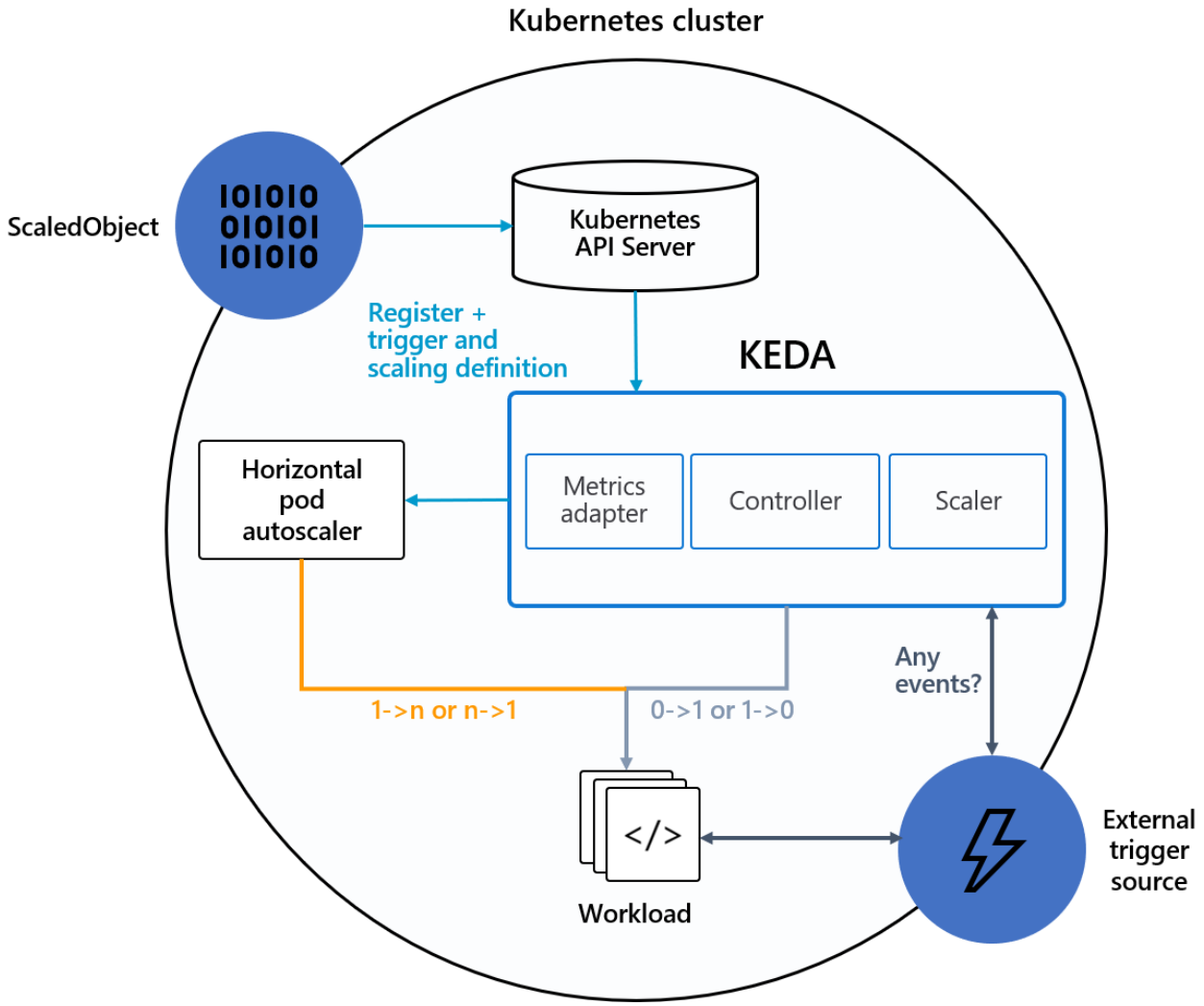


Figure 65: Kubernetes cluster

5.3.3.2 NVIDIA Triton inference server

NVIDIA Triton™ Inference Server, is an open-source inference serving software that helps standardize model deployment and execution and delivers fast and scalable AI in production.

Triton offers multi-framework and multi-model to allow a variety of workloads. These workloads typically require specific hardware (e.g., GPUs) and Triton is designed to work with many of these. To overcome the problem of unit of deployment of one, Triton has multiple options:

- Hosting multiple AI algorithms in the same container, auto balancing consumed resources between the algorithms. Although this is dynamic in execution, it requires to bundle multiple (independent) algorithms into a single container, which sometimes not preferred due to lifecycle management.
- Dynamic batching: Batching multiple requests and execution in batches usually improves throughput, but at the cost of higher latencies due to the batching.
- Concurrent execution: Triton can, depending on load and available resources, automatically scale the number of parallel executions within the container. This enables higher throughput per container, while maintaining latency at the same level.

5.3.4 Application to use case(s)

In use-case 4, the scenario of 3D reconstruction has both high compute requirements, as well as low latency requirements. The algorithm is designed to work with GPU to deliver fast results.

The workflow of 3D reconstruction can be described with the following steps:

1. Selection of the 3D reconstruction procedure on the device
2. Position the C-arm to correct location
3. Test run to test for rotation of the C-arm without touching anything (there are cables, catheters and other equipment in the room)
4. Capturing 2D Xray images while rotating the C-arm
5. Compute 3D volume based on 2D Xray images.

This is a typical use for the above-mentioned solution with pro-active scaling. As reactive scaling would take too long to start, with respect to the latency requirements. To enable pro-active scaling, an event is required by KEDA to trigger the scaling process. This event is in this case the selection of the 3D reconstruction procedure on the device.

The time between step 1 and step 5 is at least one minute, which comes from the preparation phase test run that slowly moves the C-arm. This allows KEDA to scale sufficient compute pods before the 2D Xray images arrive.

Karpenter will be used to automatically scale nodes within the cluster to fulfil all compute requirements.

NVIDIA Triton can be used to speed up the algorithm and make better use of available GPU resources.

6 Relation/interaction between solutions for safety, performance, security, and privacy

In this section, we explicitly focus on the positive relation and potential tensions between the solutions designed for improving/guaranteeing performance and safety (presented in Table 3) and the technical requirements and relevant concepts and for security and privacy.

Deliverable D3.2 (due M12) has already provided an extensive analysis on the relevant technical requirements for security and privacy, also called TSRs (in Table 8 of D3.2) which is summarized in Table 8.

All security- and privacy-related TSRs have been covered by the security and privacy concepts identified in Deliverable D3.2.

TSR No	Description	Relevant security concepts identified in D3.2 (see Table 9)
TSR 1	The TRANSACT system architecture should include protection and recovery mechanisms for data and centers for cloud services, and continuously protect data involved in transfers or transmissions	C2, C6-C15
TSR 2	The architecture should be protected against most attacks on edge computing infrastructures. This protection is mainly supposed to be against the following four categories: DDoS attacks, side-channel attacks, malware injection attacks, and authentication and authorization attacks.	C1, C5, C7, C14
TSR 3	The architecture should be protected against DDoS attacks.	C7, C14
TSR 4	The architecture should include effective solutions against flooding attacks and support the technique of detection and filtering.	C2, C7, C14, C15
TSR 5	The architecture should support packet-based detection aims to detect flooding-based attacks.	C2, C14
TSR 6	The architecture will support statistics-based approaches to detect DDoS attacks	C14
TSR 7	The architecture will be protected against zero-day attacks.	C14
TSR 8	The architecture should be protected against side-channel attacks.	C7, C14
TSR 9	The architecture will include components of a defence protection mechanisms suitable for data perturbation and differential privacy.	C5, C6, C12, C13, C14, C15
TSR 10	The architecture should be protected against malware Injection attacks.	C3, C4, C14
TSR 11	To counter the server-side injection attacks, the architecture will include detect-and-filter technique.	C14
TSR 12	The architecture will include components for defence against Device-Side Injections.	C3, C4

TSR 13	The architecture should be protected against Authentication and Authorization Attacks.	C1, C5, C6, C12, C15
TSR 14	The architecture will be protected against threats to Membership Inference Attacks.	C1, C4, C5, C6, C12
TSR 15	The architecture will be protected against Data Poisoning.	C1, C4, C5, C6, C12
TSR 16	The architecture will include components for defence against evasion attacks.	C1, C4, C6, C7, C12
TSR 17	The architecture will ensure the following security requirements: the confidentiality of permanently stored elements, executed-code authenticity, and run-time state integrity. The security architecture consists of four security mechanisms: security by separation, secure boot, secure key storage, and secure interdomain communication.	C4, C15
TSR 18	The cloud systems when used by the architecture should provide the details of how Use Case data will be handled, what types of security they already apply to the cloud infrastructure, what happens in case the system was compromised, if and how they will participate in the investigation and prosecution.	C5, C6, C9, C10, C11, C12, C15
TSR 19	The cloud systems used by the architecture should ensure that the data from the Use Cases is not shared with any third party.	C6, C8, C9, C10, C11, C12, C15
TSR 20	The cloud systems and their provider when used by the architecture should establish trust in the service offered to the Use Cases.	C8, C9, C10, C11, C12, C15
TSR 22	If used in the Use Case, an edge device in the architecture, will be secured on the basis of two factors: (1) root of trust (RoT), in which the edge device is unclonable in addition to the integrity, nonrepudiation, and authenticity of the running software at edge devices; and (2) chain of trust (CoT), in which the edge device is designed to boot up only if cryptographically signed software by a trusted entity is first executed using public-key cryptography. In addition, the keys are stored in specialized secure hardware; this hardware is also responsible for verification and RoT processes.	C1, C4, C5

Table 8: Technical security requirements (from Table 8 in Deliverable D3.2 and Section 6.9.2 of D1.2)

Concepts referred to in Table 8 are presented in Table 9.

Concept No	Concept description	Applicable to		
		Device	Edge	Cloud
C1	PKI Infrastructure	●	●	●
C2	Concept for risk analysis and management	●	●	●
C3	Runtime verification	●	●	●
C4	TPM2.0-based edge and device security	●	●	●

C5	Role-based access control rules at the business/design level	●	●	●
C6	Anonymization: prevent personal data leak		●	
C7	Security and privacy concepts for communication	●	●	
C8	Centralized Machine Learning with Decentralized Data	●	●	●
C9	Multi-cloud concept for cloud security posture management (CSPM)			●
C10	User and entity behavioural analytics (UEBA) concept for cloud security			●
C11	Cloud detection & response (Cloud DR) orchestration for multiple clouds			●
C12	Security and privacy concepts for cloud-based applications			●
C13	Safety and privacy of off-the-shelf components, including MQTT, non-doubled 4G networks, open IP networks, Unity3D			●
C14	Security and privacy concepts for secure remote driving operation	●	●	●
C15	Security and privacy requirements and patterns for the healthcare DICOM data and applications	●	●	●

Table 9: Relevant concepts for security and privacy (from Tables 6 and 7 of Deliverable D3.2)

In the rest of this section, we discuss the potential contributions of solutions developed in D3.3 to the technical security/privacy requirements and then investigate potential tensions between the concepts developed to achieve security/privacy and the performance and safety requirements.

6.1.1.1 Contributions of solutions for performance and safety on security/privacy requirements

Solutions for runtime monitoring. Detecting security threats and anomalies (such as the signature of DDoS attacks) requires monitoring the system, in particular, when the system is distributed across edge and cloud continuum. D3.3, in particular, focuses on investigating monitoring tools/techniques that are capable to gather a wide range of metrics from the system/application on the cloud and edge (S6). It also considers monitoring techniques for service continuity (availability) via S15. Thus, both S6 and S15 can be used for TSR3 to TSR6 (see Table 8).

Solutions for anomaly detection. Detecting several security attacks (such as DDoS and flooding attacks) requires analyzing output traces of the monitoring tools as these attacks leave an obvious footprint on the end-to-end response-time of the applications. Therefore, runtime solutions that use data-driven methods to predict the end-to-end response time can also be used to identify normal patterns from abnormal ones. Namely, S7 and S13 can directly contribute to achieving TSR3 to TSR6.

Some attacks involve manipulating the data (for example, those that are addressed by TSR15 or parts of the TSR11 and TSR12 that relate to data-injection attacks). S4 which provides a solution to monitor the health and performance of AI-based applications could be used to raise alarms when there is a significant change (deviation from the expected outcome/quality) of the AI applications because that could be a sign of data-injection attacks.

Solutions for providing isolation. Guaranteeing timing and performance requirements typically requires the use of isolation techniques (for example, to separate the non-real-time applications from the real-time applications when they execute on the edge or cloud). Solutions S11 and S14 provide facilities to apply such

isolation when mapping different applications to the computing resources on the cloud and edge platforms. Later in D3.5 (due in M33), more resource-management policies will be investigated which in turn improve the diversity of solutions that provide isolation. Isolation can diminish the impact of DDoS attacks on certain services/applications (hence addressing TSR2, TSR3, TSR4, TSR5, and TSR6), reduce the risk of malware injection, side-channel attacks, and authentication/authorization attacks (hence addressing TSR2, TSR8, TSR10, TSR11, TSR12, TSR13, TSR15, and TSR17).

Solutions for a combined risk analysis for safety and security risks. S12, which provides a risk-analysis framework using MARGERIT methodology, directly focuses on the assessment of the impact of security and safety risks in a use case. Therefore, it relates to all security requirements, though only some of them might be present in (or needed for) a certain use case.

The above discussions have been summarized in Table 10.

Solutions for safety and performance	**Technical security/privacy requirements**
S6 (performance observability and monitoring)	TSR3, TSR4, TSR5, TSR6 (detecting DDoS and flooding attacks)
S15 (service continuity monitoring)	TSR3, TSR4, TSR5, TSR6 (detecting DDoS and flooding attacks)
S7 (AI-based performance modeling and prediction)	TSR3, TSR4, TSR5, TSR6 (detecting DDoS and flooding attacks)
S13 (real-time machine-learning based solutions for detecting safety, security, and privacy anomalies)	TSR3, TSR4, TSR5, TSR6 (detecting DDoS and flooding attacks)
S4 (solutions for AI-monitoring)	TSR15, TSR11, TSR12 (detecting data-injection attacks)
S5 (solutions for ensuring data integrity)	TSR2, TSR8, TSR12, TSR15 (ensuring data integrity)
S11 (scenario-based performance management and reconfiguration)	TSR2, TSR3, TSR4, TSR5, TSR6, TSR8, TSR10, TSR11, TSR12, TSR13, TSR15, and TSR17 (isolating services/applications from each other)
S14 (mapping and scheduling techniques across device, edge, and cloud)	TSR2, TSR3, TSR4, TSR5, TSR6, TSR8, TSR10, TSR11, TSR12, TSR13, TSR15, and TSR17 (isolating services/applications from each other)
S12 (risk management planning/monitoring)	Relates to all security requirements

Table 10: Solutions for performance/safety that contribute to technical security/privacy requirements

6.1.1.2 Contributions of solutions/concepts for security and privacy on performance and safety requirements

Some of the concepts/solutions proposed in D3.2 (and D3.4) may not only be useful to address security requirements but also performance and safety requirements.

Concepts/solutions for runtime verification. Typically, the goal of the runtime verification (Concept C3 in Table 9) is to ensure that the safety and security requirements of a system are met. If such a mechanism is in place, it will directly contribute to the detection of safety anomalies and triggering safeguards and mode changes, resulting in guaranteeing safety and performance/timing requirements. Consequently, C3 could also be seen as an alternative way to achieve the same goal as S11, S12, S13, and S15.

Similarly, C3 can be used to detect timing anomalies (e.g., when the response-time of an application becomes much larger than expected), hence can be used along with S7 and S13 (for detecting timing anomalies) and with S11, S12, S14, S17 (to trigger resource scaling strategies to meet timing requirements).

The above discussions have been summarized in Table 11.

Concepts/solutions for security and privacy	**Safety/performance requirements**
C3 (runtime verification)	S11, S12, S13, S15 (detecting safety anomalies and triggering mode changes or safeguards)
	S7 and S13 (to detect timing anomalies)
	S11, S12, S14, S17 (to trigger resource scaling strategies to meet timing requirements)

Table 11: Concepts/solutions for security/privacy that contribute to performance/safety requirements

6.1.1.3 Tensions between solutions/concepts for security/privacy and performance/safety

Next, we will look at the tension between the concepts/solutions for security/privacy on performance/safety requirements. These tensions are summarized in Table 12:

- They may add (timing) overheads to the system (e.g., on the resource-manager component, platform, network, or application), and therefore jeopardize performance criteria such as end-to-end response time due to their overheads;
- They may increase resource contention (due to the execution of security/privacy solutions on the same computing resources);
- They may jeopardize safety or quality of service of the application (e.g., due to the use of anonymized or perturbed data which may reduce the accuracy of the application, or may require a more complex application to be implemented);

Similarly, solutions to improve/guarantee performance and safety may have the following negative impacts on the technical security/privacy requirements (this is summarized in Table 12):

- They may reduce isolation between trusted and not-trusted (or safety-critical and non-safety-critical) applications and hence increase the risk of side-channel attacks (for example if the runtime resource management strategies are security agnostic);
- They may increase the attack surface of DDoS attacks (for example, in a case where the attacker can exploit the fact that in order to “degrade performance”, it is enough to trigger the fall-back mechanism of the resource-management strategy, hence, instead of a full-fledge easy-to-detect DDoS attack, the attacker can design a more stealth attack with a small footprint);

Type of threats	Negative impact (or tension)	Concepts or solutions that may cause the impact
Performance/safety threats	Add timing overhead and hence jeopardize timing requirements	C3, C5, C6, C7, C8, C9, C11, C12, C13, C15 (due to the overhead of runtime security/privacy enforcement mechanisms)
	Increase resource contention (due to the execution of security/privacy solutions on the same computing resources)	C3, C5, C6, C7, C8, C11, C14, C15 (due to the execution of security/privacy enforcement mechanisms on the same platforms as the system)
	Jeopardize safety by impacting QoS (or functionality) of the system	C5, C6, C12, C13, C14, C15 (because of data perturbation and anonymization) C14 (because of detect-and-filter technique)
Security/privacy threats	Increase the risk of side-channel attacks	S11, S14, S17 (due to security-agnostic resource management and mapping)
	Increase the attack surface of DDoS attacks	S1, S2, S11, S14, S17 (due to providing alarms or triggering mechanisms that may result in the activation of a degraded mode)

Table 12: Tension between some of the security/privacy concepts and solutions for performance and safety

The above discussions are just an initial input for an alignment meeting of the project, where more in-depth analysis of relations and tensions between D3.3 and D3.4 will be investigated.

7 Conclusion

7.1 Summary

This deliverable has presented selected solutions for end-to-end safety and performance for distributed CPS solutions. These solutions have realized/implemented a selection of concepts identified by Task T3.1 (Deliverable D3.1) which were derived from the TRANSACT use cases and their needs, as well as the technical requirements stemming from the TRANSACT WP1 analysis (Deliverable D1.2).

The selected solutions are clustered into three main categories: application-related (Section 3), cross-cutting (Section 4), and platform-related (Section 5) solutions as introduced in Section 2. For each of these categories, a number of solution items have been developed (see a summary of these solutions in Table 3). Description of each solution item includes (i) an overview (explaining how the solution is related to the reference architecture), (ii) a discussion on the state of the art, (iii) a description of the solution and how it advances the state of the art or applies a selection of techniques from the state of the art in the context of the TRANSACT project, and (iv) a discussion on how this solution could be applied/used in different use cases or demonstrators.

This deliverable also discusses the relation and tension between solutions for performance/safety and those for security/privacy (in Section 6).

7.2 Conclusions

To achieve key objectives of the TRANSACT project, namely, *ensuring safety and performance from an end-user perspective*, this deliverable has presented a wide range of solutions focusing on improving, analyzing, or guaranteeing safety and performance of distributed safety-critical CPSs. These solutions embrace the selection of relevant concepts identified by Task T3.1 and realize these concepts in the context of specific domains (or use cases).

The final validation of these 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.

The next deliverable of Task T3.3 (namely, D3.5 which is due in month 33) aims to extend some of the solutions presented in the current deliverable (D3.3) while exploring more solutions, in particular, in the direction of runtime resource management strategies and mitigation strategies for safety and performance risks. Task T3.3 will also pay a special attention to the relation between solutions for security/privacy and solutions for performance/safety (similar to the discussion presented in Section 6).

Finally, depending on the outcome of the safety/security risk-analysis (performed by use case owners using the toolset provided by SNG), Task T3.3 will explore solutions to address the safety requirements of use cases (having various safety-integrity levels) and hence take a step towards (re)qualification of use cases.

8 References

(2022). Retrieved from POOSL toolset: https://www.poosl.org

(2022, 07 18). Retrieved from Prometheus: https://prometheus.io/

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., . . . Acharya, U. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 243-297.

Abdulkhaleq, A., & Wagner, S. (2015). XSTAMPP: an eXtensible STAMP platform as tool support for safety engineering. *2015 STAMP Workshop.* Boston: MIT. Retrieved from https://elib.uni-stuttgart.de/handle/11682/3550

Aceto, G., Botta, A., De Donato, W., & Pescapè, A. (2013). Cloud monitoring: A survey. *Computer Networks, 57*(9), 2093-2115.

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.

Akesson, B., Nasri, M., Nelissen, G., Altmeyer, S., & Davis, R. I. (2020). An Empirical Survey-based Study into Industry Practice in Real-time Systems. *IEEE Real-Time Systems Symposium (RTSS)*, (pp. 1–9).

Alarifi, A., Al-Salman, A., Alsaleh, M., Alnafessah, A., Al-Hadhrami, S., Al-Ammar, M. A., & Al-Khalifa, H. S. (2017). Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances. *Sensors*.

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.

Alshammari, M. M., Alwan, A. A., Nordin, A., & Al-Shaikhli, I. F. (2017). Disaster recovery in single-cloud and multi-cloud environments: Issues and challenges. *4th IEEE International Conference on Engineering Technologies and Applied Sciences* (pp. 1-17). IEEE.

Al-Zoubi, K., & Wainer, G. (2020). Modelling Fog & Cloud Collaboration Methods on Large Scale. *Winter Simulation Conference (WSC)*, (pp. 2161-2172).

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

Amini, A., Schwarting, W., Soleimany, A., & Rus, D. (2020). Deep Evidential Regression. *Advances in Neural Information Processing Systems*, 14927-14937.

Amiri, M., & Mohammad-Khanli, L. (2017). Survey on prediction models of applications for resources provisioning in cloud. *Journal of Network and Computer Applications*, 93-113.

Andersson, J. L. (2009). Modeling dimensions of self-adaptive software systems. In *Software engineering for self-adaptive systems* (pp. 27-47). Springer.

Arcaini, P. R. (2015). Modeling and analyzing MAPE-K feedback loops for self-adaptation. *10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems* (pp. 13-23). IEEE.

Ardagna, D., Barbierato, E., Gianniti, E., Gribaudo, M., Pinto, T., da Silva, A., & Almeida, J. M. (2021). Predicting the performance of big data applications on the cloud. *The Journal of Supercomputing*, 1321-1353.

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

Audsley, N., Burns, A., Richardson, M., Tindell, K., & Wellings, A. J. (1993). Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling. *Software Engineering Journal, 8*, 284–292.

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.

Baccour, N., Koubâa, A., Noda, C., Fotouhi, H., Alves, M., Youssef, H., . . . Puccinelli, D. a. (2013). External Radio Interference. In N. Baccour, A. Koubâa, C. Noda, H. Fotouhi, M. Alves, H. Youssef, . . . D. a. Puccinelli, *Radio Link Quality Estimation in Low-Power Wireless Networks.*

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

Baker, T. P. (2003). Multiprocessor EDF and deadline monotonic schedulability analysis. *IEEE Real-Time Systems Symposium (RTSS)*, (pp. 120–129).

Baker, T. P., & Cirinei, M. (2007). Brute-Force Determination of Multiprocessor Schedulability for Sets of Sporadic Hard-Deadline Tasks. *OPODIS*, (pp. 62–75).

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

Balarin, F., Chiodo, M., Giusto, P., Hsieh, H., Jurecska, A., Lavagno, L., . . . Tabbara, B. (1997). *Hardware-Software Co-design of Embedded Systems: The POLIS Approach.* Springer New York.

Basu, A., Bozga, M., & Sifakis, J. (2006). Modeling Heterogeneous Real-time Components in BIP. *4th IEEE International Conference on Software Engineering and Formal Methods, SEFM'06.* IEEE.

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.

Bellorado, J., Ghassemzadeh, S. S., Greenstein, L. J., & Sveinsson, T. a. (2003). Coexistence of Ultra-Wideband Systems with IEEE-802.11a Wireless LANs. *GLOBECOM'03. IEEE Global Telecommunications Conference* , (pp. 410-414).

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.

Bhargavan, K. C. (2001). What packets may come: automata for network monitoring. *ACM SIGPLAN Notices, 36(3)*, 206-219.

Bijlsma, T., & Hendriks, T. (2017). A fail-operational truck platooning architecture. *IEEE Intelligent Vehicles Symposium (IV)* (pp. 1819-1826). Redondo Beach, CA: IEEE. doi:https://doi.org/10.1109/IVS.2017.7995970

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.

Box, D. (1998). *Essential COM.* Addison-Wesley.

Brown, J., Roedig, U., Boano, C. A., & Römer, K. (2014). Estimating Packet Reception Rate in Noisy Environments. *39th Annual IEEE Conference on Local Computer Networks Workshops*, (pp. 583-591).

Brunner, H., Stocker, M., Schuh, M., Schuß, M., Boano, C. A., & Römer, K. (2022). Understanding and Mitigating the Impact of Wi-Fi 6E Interference on Ultra-Wideband Communications and Ranging. *2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, (pp. 92-104).

Buchgeher, G., Ramler, R., Stummer, H., & Kaufmann, H. (2021). Adopting Microservices for Industrial Control Systems: A Five Step Migration Path. *International Conference on Emerging Technologies and Factory Automation (ETFA)* (pp. 1-8). IEEE.

Bulej, L., Bures, T., Filandr, A., Hnetynka, P., Hnetynkova, I., Pacovsky, J., . . . Gerostathopoulos, I. (2021). Managing latency in edge--cloud environment. *Journal of systems and software*.

Burmyakov, A., Bini, E., & Lee, C.-G. (2022). Towards a Tractable Exact Test for Global Multiprocessor Fixed Priority Scheduling. *IEEE Transactions on Computers*. doi:10.1109/TC.2022.3142540

Burmyakov, A., Bini, E., & Tovar, E. (2015). An exact schedulability test for global FP using state space pruning. *RTNS*, (pp. 225–234).

Buttazzo, G. (2011). *Hard real-time computing systems: Predictable scheduling algorithms and applications.* Springer.

Buttazzo, G. C. (2011). *Hard real-time computing systems: predictable scheduling algorithms and applications* (Third ed.). Springer Science & Business Media.

Cal, P., Wang, H., Huang, H., Liu, Y., & Liu, M. (2021). Vision-based autonomous car racing using deep imitative reinforcement learning. *IEEE Robotics and Automation Letters*, 7262-7269.

Calheiros, R. N., Masoumi, E., Ranjan, R., & Buyya, R. (2014). Workload prediction using ARIMA model and its impact on cloud applications' QoS. *IEEE transactions on cloud computing* (pp. 449-458). IEEE.

Calheiros, R. N., Ranjan, R., Beloglazov, A., Rose, C. A., & Buyya, R. (2011). CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software: Practice and Experience (SPE)*, 23-50.

Casini, D., Biondi, A., Nelissen, G., & Buttazzo, G. (2018). Partitioned Fixed-Priority Scheduling of Parallel Tasks Without Preemptions. *2018 IEEE Real-Time Systems Symposium (RTSS)*, (pp. 421-433). doi:10.1109/RTSS.2018.00056

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.

Chong, A., & Song, C. (2019). A Framework For The Continuous Calibration Of Building Energy Models With Uncertainty. *Proceedings of the 16th IBPSA Conference*, (pp. 4562-4569).

Cirillo, F. S. (2019). A standard-based open source IoT platform: FIWARE. *IEEE Internet of Things Magazine*, 12-18.

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

Craciunas, S. a. (2016). Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks. *International Conference on Real-Time Networks and Systems (RTNS)*, (pp. 183-192).

Crankshaw, D., Sela, G.-E., Mo, X., Zumar, C., Stoica, I., Gonzalez, J., & Tumanov, A. (2020). InferLine: latency-aware provisioning and scaling for prediction serving pipelines. *Proceedings of the 11th ACM Symposium on Cloud Computing* (pp. 477-491). ACM.

Cuomo, A., Rak, M., & Villano, U. (2015). Performance prediction of cloud applications through benchmarking and . *International Journal of Computational Science and Engineering*, 46-55.

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.

Davis, R. I., Burns, A., Bril, R. J., & Lukkien, J. J. (2007). Controller Area Network (CAN) schedulability analysis : refuted, revisited and revised. *Real-Time Systems, 35*, 239–272.

D-Cube. (n.d.). Retrieved from D-Cube: https://iti-testbed.tugraz.at/wiki/index.php/Main_Page

de Alfaro, L., & Henzinger, T. (2001). Interface Theories for Component-Based Design. *Embedded Software, EMSOFT 2001* (pp. 148–165). Springer.

de Moura, L., & Bjørner, N. (2008). Z3: An Efficient SMT Solver. *Tools and Algorithms for the Construction and Analysis of Systems (TACAS).* Berlin: Springer.

Deequ. (2022). Retrieved from Deequ - Unit Tests for Data: https://github.com/awslabs/deequ

Der Kiureghian, A., & Ditlevsen, O. (2009). Aleatory or epistemic? Does it matter? *Structural safety*, 105-112.

Derler, P., Lee, E., & Vincentelli, A. S. (2012). Modeling cyber–physical systems. *Proceedings of the IEEE*.

Dong, Z., & Liu, C. (2019). Work-in-progress: Non-preemptive scheduling of sporadic gang tasks on multiprocessors. *Work-in-Progress of IEEE Real-Time Systems Symposium (WiP-RTSS)* (pp. 512–515). IEEE.

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

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). *Microservices: Yesterday, Today, and Tomorrow.* Cham: Springer International Publishing.

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

Eclipse TRACE4CPS. (2022). Retrieved from https://www.eclipse.org/trace4cps/

Eisenbrand, F. a. (2008). Static-priority real-time scheduling: Response time computation is NP-hard. *Real-Time Systems Symposium (RTSS)* (pp. 397 - 406). IEEE.

Eisenbrand, F. a. (2010). EDF-schedulability of synchronous periodic task systems is coNP-hard. *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (pp. 1029 - 1034). Society for Industrial and Applied Mathematics.

Eker, J., Janneck, J. W., Lee, E. A., Liu, J., Liu, X., Ludvig, J., . . . Xiong, a. Y. (2003). Taming Heterogeneity - the Ptolemy Approach. *Proceedings of the IEEE, 91*(1).

Endo, P., Santos, G., Rosendo, D., Gomes, D., Moreira, A., Kelner, J., . . . Mahloo, M. (2017). Minimizing and managing cloud failures. *Computer, 50*(11), 86-90.

Envoy. (2022, 07 20). Retrieved from https://www.envoyproxy.io/

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

Etemad, M., Aazam, M., & St-Hilaire, M. (2017). Using DEVS for modeling and simulating a Fog Computing environment. *International Conference on Computing, Networking and Communications (ICNC)* (pp. 849-854). IEEE.

Evidently AI. (2022). Retrieved from Open-source Machine Learning Monitoring: https://evidentlyai.com

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.

Feiertag, N., Richter, K., Nordlander, J., & Jonsson, J. (2009). A compositional framework for end-to-end path delay calculation of automotive systems under different path semantics. *Real-Time Systems Symposium (RTSS).*

Feitelson, D. G., & Rudolph, L. (1992). Gang scheduling performance benefits for fine-grain synchronization. *Journal of Parallel and Distributed Computing, 16*, 306–318.

Ferdinand Brasser, K. B. (2016). Remote attestation for low-end embedded devices: The prover's perspective. *Proceedings of the Design Automation Conference.*

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.

Firoozbakhsh, B., Pratt, T. G., & Jayant, N. (2003). Analysis of IEEE 802.11a Interference on UWB Systems. *IEEE Conference on Ultra Wideband Systems and Technologies, 2003*, (pp. 473-477).

Fitzgerald, F., Larsen, P., & Verhoef, M. (2014). *Collaborative Design for Embedded Systems.* Springer-Verlag.

Friedberg, I., McLaughlin, K., Smith, P., Laverty, D., & Sezer, S. (2017). STPA-SafeSec: Safety and security analysis for cyber-physical systems. *ournal of information security and applications, 34*, 183-196.

Frydrychowicz, A. (2021). Automatic, log file-based process analysis of a clinical 1.5T MR scanner: a proof-of-concept study. *RoFo : Fortschritte auf dem Gebiete der Rontgenstrahlen und der Nuklearmedizin*, 919-927.

Fuh, C.-D., & Tartakovsky, A. G. (2018). Asymptotic Bayesian theory of quickest change detection for hidden Markov models. *IEEE Transactions on Information Theory* (pp. 511-529). IEEE.

Fürst, S., Mössinger, J., Bunzel, S., Weber, T., Kirschke-Biller, F., Heitkämper, P., . . . Lange, K. (2009). AUTOSAR–A Worldwide Standard is on the Road. *14th International VDI Congress Electronic Systems for Vehicles, Baden-Baden*, *62*, p. 5.

Gal, Y., & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *International conference on machine learning* (pp. 1050-1059). PMLR.

Gan, Y., & et al. (2019). An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud & Edge Systems. *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '19)* (pp. 3-18). New York: Association for Computing Machinery.

Gan, Y., Zhang, Y., Cheng, D., Shetty, A., Rathi, P., Katarki, N., . . . Delimitrou, C. (2019, April). An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud & Edge Systems. *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (pp. 3–18). New York, NY, USA: Association for Computing Machinery. doi:10.1145/3297858.3304013

Gartziandia, A., Ayerdi, J., Arrieta, A., Ali, S., Yue, T., Agirre, A., . . . Arratibel, M. (2021). Microservices for Continuous Deployment, Monitoring and Validation in Cyber-Physical Systems: an Industrial Case Study for Elevators Systems. *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*, (pp. 46-53).

Geilen, M., Basten, T., Theelen, B., & Otten, R. (2007). An algebra of Pareto points. *Fundamenta Informaticae*, 35-74.

Geilen, M., Tripakis, S., & Wiggers, M. (2011). The Earlier the Better: A Theory of Timed Actor Interfaces. *14th Int. Conf. on Hybrid Systems: Computation and Control, HSCC '11.* ACM.

Gohari, P., Nasri, M., & Voeten, J. (2022). Data-Age Analysis for Multi-Rate Task Chains under Timing Uncertainty. *Conference on Real-Time Networks and Systems (RTNS)*, (pp. 24-35).

Goniwada, S. R. (2022). Observability. In S. R. Goniwada (Ed.), *Cloud Native Architecture and Design: A Handbook for Modern Day Architecture and Design with Enterprise-Grade Examples* (pp. 661–676). Berkeley, CA: Apress. doi:10.1007/978-1-4842-7226-8_19

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

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

Goossens, J. P. (2019). ACCEPTOR: a model and a protocol for real-time multi-mode applications on reconfigurable heterogeneous platforms. *27th International Conference on Real-Time Networks and Systems*, (pp. 209-219).

Grafana. (2022, 07 18). Retrieved from https://grafana.com/

Grafana Mimir. (2022, 07 20). Retrieved from https://grafana.com/oss/mimir/

Graphite. (2022, 07 20). Retrieved from https://github.com/graphite-project/graphite-web

Great expectations. (2022). Retrieved from Great expectations: Always know to expect from your data: https://github.com/great-expectations/great_expectation

Gribaudo, M., Iacono, M., & Manini, D. (2017). Performance Evaluation Of Massively Distributed Microservices Based Applications. *31st European Conference on Modelling and Simulation.*

Großwindhager, B., Boano, C. A., Rath, M., & Römer, K. (2018). Enabling Runtime Adaptation of Physical Layer Settings for Dependable UWB Communications. *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, (pp. 1-11).

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

Group, T.-S. N. (2016). *Official Website of the 802.1 Time-Sensitive Networking Task Group.* IEEE.

Guan, N., Gu, Z., Deng, Q., Gao, S., & Yu, G. (2007). Exact Schedulability Analysis for Static-Priority Global Multiprocessor Scheduling Using Model-Checking. *SEUS*, (pp. 263–272).

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).

Günal, M. a. (2010). Discrete event simulation for performance modelling in health care: a review of the literature. *Journal of simulation*, 42-51.

Gunn, M. (2017). Improving MRI scanner utilization using modality log files. *Journal of the American college of radiology*, 783-786.

Hamann, A., Dasari, D., Wurst, F., Sañudo, I., Capodieci, N., Burgio, P., & Bertogna, M. (2019). WATERS Industrial Challenge 2019. Retrieved from http://archives.ecrts.org/fileadmin/WebsitesArchiv/ecrts2019/waters/waters-industrial-challenge/index.html

Hause, M. (2006). The SysML Modelling Language. *Fifteenth European Systems Engineering Conference.*

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.

Heemels, M., & Muller, G. (2006). *Boderc: Model-based design of high-tech systems.* Embedded Systems Institute.

Heinrich, R., van Hoorn, A., Knoche, H., Li, F., Lwakatare, L. E., Pahl, C., . . . Wettinger, J. (2017). Performance Engineering for Microservices: Research Challenges and Directions. *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion* (pp. 223-226). New York: Association for Computing Machinery.

Heinrich, R., van Hoorn, A., Knoche, H., Li, F., Lwakatare, L. E., Pahl, C., . . . Wettinger, J. (2017, April). Performance Engineering for Microservices: Research Challenges and Directions. *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion* (pp. 223–226). New York, NY, USA: Association for Computing Machinery. doi:10.1145/3053600.3053653

Hendriks, M., & Basten, T. (2018). Performance Engineering with Trace. *Bits & Chips*.

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

Huang, P., Guo, C., Zhou, L., Lorch, J. R., Dang, Y., Chintalapati, M., & Yao, R. (2017). Gray failure: The achilles' heel of cloud-scale systems. *Proceedings of the 16th Workshop on Hot Topics in Operating Systems* (pp. 150-155). Whistler, BC: ACM.

Hullermeier, E., & Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 457-506.

Idziorek, J. (2010). Discrete event simulation model for analysis of horizontal scaling in the cloud computing model. *Proceedings of the Winter Simulation Conference (WSC '10)*, (pp. 3004–3014).

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.*

Inc., G. (n.d.). *CP-SAT Solver Guide. Google Inc.* http://developers.google. com/optimization/cp/cp_solver.

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.

Istio. (2022, 07 20). Retrieved from https://istio.io/

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

Jaeger. (2022, 07 18). Retrieved from https://www.jaegertracing.io/

Jamshidi, P., Pahl, C., Mendonça, N. C., Lewis, J., & Tilkov, S. (2018). Microservices: The Journey So Far and Challenges Ahead. *IEEE Software*, 24-35.

Jansen, M., Al-Dulaimy, A., Papadopoulos, A. V., Trivedi, A., & Iosup, A. (2022). *The SPEC-RG Reference Architecture for the Edge Continuum.* doi:10.48550/arXiv.2207.04159

Jiang, J., Lu, J., Zhang, G., & Long, G. (2013). Optimal Cloud Resource Auto-Scaling for Web Applications. *13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, (pp. 58-65).

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.

Karim El Defrawy, D. P. (2012). SMART: Secure and Minimal Architecture for (Establishing Dynamic) Root of Trust. . *In Proceedings of Network and Distributed System Security Symposium*, (pp. 1-15).

Kavallieratos, G., Katsikas, S., & Gkioulos, V. (2020). Cybersecurity and safety co-engineering of cyberphysical systems—a comprehensive survey. *Future Internet,, 12*(4).

Kaynar, D., Lynch, N., Segala, R., & Vaandrager, F. (2003). Timed I/O Automata: A Mathematical Framework for Modeling and Analyzing Real-Time Systems. *24th IEEE International Real-Time Systems Symposium, RTSS '03.* IEEE.

Kennedy, M., & O'Hagan, A. (2001). Bayesian calibration of computer models. *Royal Statistical Society*, 425–464.

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.

Kim, J. L. (2014). OpenIoT: An open service framework for the Internet of Things. *IEEE world forum on internet of things (WF-IoT)* (pp. 89-93). IEEE.

Koen Tange, M. D. (2020). A systematic survey of industrial Internet of Things security: Requirements and fog computing opportunities. *IEEE Communications Surveys & Tutorials*, 2489–2520.

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).

Kramer, S., Ziegenbein, D., & Hamann, A. (2015). Real world automotive benchmarks for free. *Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS).*

Kubernetes. (2014, 7 6). Retrieved 7 1, 2022, from https://kubernetes.io/

Kumar, J., & Singh, A. (2018). Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Generation Computer Systems* (pp. 41-52). Elsevier.

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*.

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, E. (2017). *Plato and the Nerd - The Creative Partnership of Humans and Technology.* MIT Press.

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*.

Levenson, N. (2019). *Leveson, N. (2019). An Improved Design Process for Complex, Control-Based Systems Using STPA and a Conceptual Architecture.* Boston, MA: Massachusetts Institute of Technology.

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

Li, B., Peng, X., Xiang, Q., Wang, H., Xie, T., Sun, J., & Liu, X. (2021, November). Enjoy your observability: an industrial survey of microservice tracing and analysis. *Empirical Software Engineering, 27*, 25. doi:10.1007/s10664-021-10063-9

Li, W., Lemieux, Y., Gao, J., Zhao, Z., & Han, Y. (2019, April). Service Mesh: Challenges, State of the Art, and Future Research Opportunities. *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, (pp. 122–1225). doi:10.1109/SOSE.2019.00026

Lin, S., Miao, F., Zhang, J., Zhou, G., Gu, L., He, T., . . . Pappas, G. J. (2006). ATPC: Adaptive Transmission Power Control for Wireless Sensor Networks. *Proceedings of the 4th ACM International Conference* (pp. 223-236). ACM.

Linkerd. (2022, 07 20). Retrieved from https://linkerd.io/

Lisova, E., Šljivo, I., & Čaušević, A. (2018). Safety and security co-analyses: A systematic literature review. *EEE Systems Journal, 13*(3), 2189-2200.

Liu, C. L. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of ACM*, 46 - 61.

LTTng. (2022). *LTTng. An open source tracing framework for Linux*. Retrieved from https://lttng.org/

Lynch, N., & Tuttle, M. (1987). Hierarchical Correctness Proofs for Distributed Algorithms. *6th Annual ACM Symposium on Principles of Distributed Computing, PODC '87.* ACM.

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

Mapillary. (2021). *OpenSFM*. Retrieved from https://opensfm.org/

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

Medvidovic, N., Rosenblum, D. S., Redmiles, D. F., & Robbins, J. E. (2002). Modeling software architectures in the Unified Modeling Language. *ACM ToSEM*, 2-57.

Melani, A., Bertogna, M., Bonifaci, V., Marchetti-Spaccamela, A., & Buttazzo, G. C. (2015). Response-Time Analysis of Conditional DAG Tasks in Multiprocessor Systems. *ECRTS*, (pp. 211–221).

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

Mielczarek, B. (2012). Application of computer simulation modeling in the health care sector: a survey. *Simulation*, 197-216.

MOHAMMADREZA BARZEGARAN, A. C. (2020). Performance optimization of control applications on fog computing platforms using scheduling and isolation. *IEEE ACCESS*, 104085-104098.

Naghshnejad, M., & Singhal, M. (2020). A hybrid scheduling platform: a runtime prediction reliability aware scheduling platform to improve HPC scheduling performance. *The Journal of Supercomputing, 76*.

Nasri, M., & Brandenburg, B. B. (2017). An Exact and Sustainable Analysis of Non-preemptive Scheduling. *RTSS*, (pp. 12-23).

Nasri, M., Nelissen, G., & Brandenburg, B. B. (2018). A Response-Time Analysis for Non-Preemptive Job Sets under Global Scheduling. *ECRTS*, (pp. 9:1–9:23).

Nasri, M., Nelissen, G., & Brandenburg, B. B. (2019). Response-time analysis of limited-preemptive parallel DAG tasks under global scheduling. *ECRTS*, (pp. 21:1–21:23).

Nelis, V. G. (2009). Two protocols for scheduling multi-mode real-time systems upon identical multiprocessor platforms. *21st Euromicro Conference on Real-Time Systems* (pp. 151-160). IEEE.

Nelissen, G., Marce-i Igual, J., & Nasri, M. (2022). Response-Time Analysis for Non-Preemptive Periodic Moldable Gang Tasks. *ECRTS*, (pp. 12:1–12:22).

Netsch, T. a. (2019). *Delayed reconstruction for MRI and CT.* Technical Note PR-TN 2019/00268, Philips Research.

Nguyen, M., Alesawi, S., Li, N., Che, H., & Jiang, H. (2020). A black-box Fork-join latency prediction model for data-intensive applications. *IEEE Transactions on Parallel and Distributed Systems* (pp. 1983-2000). IEEE.

Niklas Reusch, S. S. (2022). Dependability-Aware Routing and Scheduling for Time-Sensitive Networking. *IET Cyber-Physical Systems: Theory & Applications*.

Noda, C., Prabh, S., Alves, M., Boano, C. A., & Voigt, T. (2011). Quantifying the Channel Quality for Interference-Aware Wireless Sensor Networks. *ACM SIGBED Review*, 43-48.

Object Management Group. (2012). *The Common Object Request Broker: Architecture and Specification Version 3.3.*

Office of Government Commerce (OGC). (2007). *ITIL Service Design.* United Kingdom: The Stationery Office.

Olena Skarlat, V. K. (2018). A Framework for Optimization, Service Placement, and Runtime Operation in the Fog. *11th International Conference on Utility and Cloud Computing (UCC)* (pp. 164-173). IEEE.

OpenCensus. (2022, 07 20). Retrieved from https://opencensus.io/

OpenTelemetry. (2022, 07 20). Retrieved from https://opentelemetry.io/

OpenTracing. (2022, 07 20). Retrieved from https://opentracing.io/

OSGi Alliance. (2018). *OSGi release 7.*

Pang, B., T'Jonck, K., Claeys, T., Pissoort, D., Hallez, H., & Boydens, J. (2021). Bluetooth Low Energy Interference Awareness Scheme and Improved Channel Selection Algorithm for Connection Robustness. *Sensors*.

Parappurath, V., Voeten, J., & Kotterink, K. (2013). Calibration error bound estimation in performance modeling. *2013 Euromicro Conference on Digital System Design*, (pp. 97-102).

Park, E., Lee, M.-S., & Bahk, S. (2019). AdaptaBLE: Data Rate and Transmission Power Adaptation for Bluetooth Low Energy. *2019 IEEE Global Communications Conference (GLOBECOM)*, (pp. 1-6).

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.

Parry, G. W. (1996). The characterization of uncertainty in probabilistic risk assessments of complex systems. *Reliability Engineering & System Safety*, 119-126.

Paul Pop, B. Z. (2021). The FORA Fog Computing Platform for Industrial IoT. *Information Systems*, 101727.

Percepio. (2022). *Percepio Tracealyzer. Stop Guessing with Visual Trace Diagnostics*. Retrieved from https://percepio.com/tracealyzer/

Pimentel, A., Thompson, M., Polstra, S., & Erbas, C. (2008). Calibration of Abstract Performance Models for System-Level Design Space Exploration. *Journal of Signal Processing Systems* , 99-114.

Pirc, P. (2021). *UWB Explained: What Secure Ranging Means for the IoT and Beyond*. Retrieved from everythingRF: https://www.everythingrf.com/community/uwb-explained-what-secure-ranging-means-for-the-iot-and-beyond

Poirot, V., & Landsiedel, O. (2021). eAFH: Informed Exploration for Adaptive Frequency Hopping in Bluetooth Low Energy. *arXiv preprint arXiv:2112.03046*.

Prokhorenko, V., & Babar, M. A. (2020). Architectural Resilience in Cloud, Fog and Edge Systems. *IEEE Access*, 28078-28095.

Prometheus. (2012, 11 24). Retrieved 7 19, 2022, from https://prometheus.io/

Qiu, H., Banerjee, S. S., Jha, S., Kalbarczyk, Z. T., & Iyer, R. K. (2020). FIRM: An Intelligent Fine-grained Resource Management Framework for SLO-Oriented Microservices. *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)* (pp. 805-825). USENIX.

Rahman, P., & Lama, J. (2019). Predicting the End-to-End Tail Latency of Containerized Microservices in the Cloud. *IEEE International Conference on Cloud Engineering (IC2E), 2019*, (pp. 200-210).

Ranschaert, E. (2021). Optimization of radiology workflow with artificial intelligence. *Radiologic clinics of North America*, 955-966.

Rapita Systems. (2022). *Rapita Systems - RapiTask Zero*. Retrieved from https://www.rapitasystems.com/products/rapitaskzero

Real, J. C. (2004). Mode change protocols for real-time systems: A survey and a new proposal. *Real-time systems, 26(2)*, 161-197.

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

Rincon Vija, F. A., Cregut, S., Papadopoulos, G. Z., & Montavont, N. (2021). From Wired to Wireless BMS in Electric Vehicles. *17th International Conference on Mobility, Sensing and Networking (MSN)*, (pp. 255-262).

Rodrigo Vieira Steiner, E. L. (2016). Attestation in Wireless Sensor Networks: A Survey. *ACM Computing Surveys*, 1–31.

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

RTEMS. (2022). *RTEMS tracing framework*. Retrieved from https://docs.rtems.org/branches/master/user/tracing/index.html

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

Saleh Sedghpour, M. R., Klein, C., & Tordsson, J. (2022, April). An Empirical Study of Service Mesh Traffic Management Policies for Microservices. *Proceedings of the 2022 ACM/SPEC on International Conference on Performance Engineering* (pp. 17–27). New York, NY, USA: Association for Computing Machinery. doi:10.1145/3489525.3511686

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.

Sangiovanni-Vincentelli, A., Damm, W., & Passerone, R. (2012). Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems. *European Journal of Control, 18*(3).

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.

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

Seo, C., Zeigler, B. P., & Kim, D. (2018). DEVS markov modeling and simulation: formal definition and implementation. *Proceedings of the Theory of Modeling and Simulation Symposium (TMS '18)*, (pp. 1-12).

Serrano, M. A., Melani, A., Bertogna, M., & Quiñones, E. (2016). Response-time analysis of DAG tasks under fixed priority scheduling with limited preemptions. *DATE*, (pp. 1066–1071).

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.

Sigelman, B. H., Barroso, L. A., Burrows, M., Stephenson, P., Plakal, M., Beaver, D., . . . Shanbhag, C. (2010). *Dapper, a Large-Scale Distributed Systems Tracing Infrastructure.* Tech. rep., Google, Inc. Retrieved March 16, 2022, from https://research.google.com/archive/papers/dapper-2010-1.pdf

Spörk, M., Boano, C. A., & Römer, K. (2019). Improving the Timeliness of Bluetooth Low Energy in Noisy RF Environments. *EWSN*, (pp. 23-34).

Spörk, M., Classen, J., Boano, C. A., & Römer, K. (2020). Improving the Reliability of Bluetooth Low Energy Connections. *EWSN*, (pp. 144-155).

Spörk, M., Schuß, M., Boano, C. A., & Römer, K. (2021). Ensuring End-to-End Dependability Requirements in Cloud-based Bluetooth Low Energy Applications. *EWSN*, (pp. 55-66).

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.

Steiner, S. S. (2016). Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks. *RTNS '16: Proceedings of the 24th International Conference on Real-Time Networks and Systems* (pp. 1-10). ACM.

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

Stocker, M., Großwindhager, B., Boano, C. A., & Römer, K. (2020). Towards Secure and Scalable UWB-based Positioning Systems. *2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, (pp. 247-255).

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).

Sun Microsystems. (1997). *Sun Microsystems. JavaBeans.*

Sun, Y., & Lipari, G. (2016). A Pre-Order Relation for Exact Schedulability Test of Sporadic Tasks on Multiprocessor Global Fixed-Priority Scheduling. *Real-Time Systems, 52*, 323–355.

Swedberg, C. (2020). *FCC's Wi-Fi 6 GHz Plan Poses Interference for UWB*. Retrieved from RFID Journal: https://www.rfidjournal.com/fccs-wi-fi-6-ghz-plans-poses-interference-for-uwb

T.Pereira, L. A. (2017). Network and information security challenges within Industry 4.0 paradigm. *Procedia Manufacturing*, 1253-1260.

Tensorflow. (2015, 11 9). Retrieved 7 1, 2022, from https://www.tensorflow.org/

Theelen, B. (2004). *Performance Modelling for System-Level Design.* Eindhoven University of Technology.

Thramboulidis, K., Vachtsevanou, D. C., & Solanos, A. (2018). Cyber-physical microservices: An IoT-based framework for manufacturing systems. *International Conference on Industrial Cyber-Physical Systems (ICPS)* (pp. 232-239). IEEE.

TRANSACT EU project. (2022, 05 30). *D8 (D3.1) Selection of concepts for end-to-end safety and performance for distributed CPS solutions.* (T. Hendriks, Ed.) Retrieved from https://transact-ecsel.eu/: https://transact-ecsel.eu/wp-content/uploads/2022/06/D3.1-Selection-of-concepts-for-end-to-end-safety-and-performance-for-distributed-CPS-solutions.pdf

Turin, G., Borgarelli, A., Donetti, S., Johnsen, E., Tapia Tarifa, S., & Damiani, F. (2020). A Formal Model of the Kubernetes Container Framework. *ISoLA 2020.* Springer.

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.

Urgaonkar, B., Shenoy, P., Chandra, A., Goyal, P., & Wood, T. (2008). Agile dynamic provisioning of multi-tier internet applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1-39.

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

Vasileios Karagiannis, A. P. (2017). Network-Integrated Edge Computing Orchestrator for Application Placement. *13th International Conference on Network and Service Management (CNSM)* (pp. 1-5). IEEE.

Voeten, J. P. (1995). *POOSL : an object-oriented specification language for the analysis and design of hardware/software systems.* Eindhoven University of Technology.

Wandeler, E., & Thiele, L. (2005). Real-time Interfaces for Interface-based Design of Real-time Systems with Fixed Priority Scheduling. *5th ACM International Conference on Embedded Software, EMSOFT '05.* ACM.

Wang, H., & Yeung, D.-Y. (2016). Towards Bayesian deep learning: A framework and some existing methods. *IEEE Transactions on Knowledge and Data Engineering* (pp. 3395-3408). IEEE.

Ward, R., Choudhary, R., Gregory, A., Jans-Singh, M., & Girolami, M. (2021). Continuous calibration of a digital twin: Comparison of particle filter and Bayesian calibration approaches. *Data-Centric Engineering*.

Wei Jiang, P. P. (2017). Design optimization for security-and safety-critical distributed real-time applications. *Microprocessors and Microsystems*, 401-415.

Weißbach, M. S. (2017). Coordinated execution of adaptation operations in distributed role-based software systems. *Symposium on Applied Computing*, (pp. 45-50).

Weyns, D. S. (2013). On patterns for decentralized control in self-adaptive systems. In *Software Engineering for Self-Adaptive Systems II* (pp. 76-107). Springer.

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.

Wyss, M. a. (2021). Secure and Scalable QoS for Critical Applications. *International Symposium on Quality of Service (IWQOS)* (pp. 1-10). IEEE/ACM.

Wyss, M. a. (2021). Secure and Scalable QoS for Critical Applications. *International Symposium on Quality of Service (IWQOS)* (pp. 1-10). IEEE/ACM.

Xavier Carpent, N. R. (2018). Remote attestation of IoT devices via SMARM: Shuffled measurements against roving malware. *IEEE International Symposium on Hardware Oriented Security and Trust*, (pp. 9-16).

Yalcinkaya, B., Nasri, M., & Brandenburg, B. B. (2019). An Exact Schedulability Test for Non-Preemptive Self-Suspending Real-Time Tasks. *DATE*, (pp. 1228-1233).

Yan, M., Liang, X., Lu, Z., Wu, J., & Zhang, W. (2021). HANSEL: adaptive horizontal scaling of microservices using Bi-LSTM. *Applied Soft Computing.* Elsevier.

Yomsi, P. M. (2010). Scheduling multi-mode real-time systems upon uniform multiprocessor platforms. *15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)* (pp. 1-8). IEEE.

Young, W., & Leveson, N. (2013). Systems thinking for safety and security. *Proceedings of the 29th Annual Computer Security Applications Conference*, (pp. 1-8). New Orleans.

Yu, J., Wagner, S., & Luo, F. (2021). Data-flow-based adaption of the System-Theoretic Process Analysis for Security. *PeerJ Computer Science, 7*(e362).

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

Zeng, R., Hou, X., Z. L., Li, C., Zheng, W., & Guo, M. (2022). Performance optimization for cloud computing systems in the microservice era: state-of-the-art and research opportunities. *Frontiers of Computer Science*.

Zhang, L., Zhou, W.-D., Chang, P.-C., Yang, J.-W., & Li, F.-Z. (2013). Iterated time series prediction with multiple support vector regression models. *Neurocomputing* (pp. 411-422). Elsevier.

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

Zhang, Y., Gan, Y., & Delimitrou, C. (2019). µqSim: Enabling Accurate and Scalable Simulation for Interactive Microservices. *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, (pp. 212-222).

Zhang, Y., Hua, W., Zhou, Z., Suh, G., & Delimitrou, C. (2021). Sinan: ML-based and QoS-aware resource management for cloud microservices. *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, (pp. 167-181).

Zhou, X., Peng, X., Xie, T., Sun, J., Ji, C., Li, W., & Ding, D. (2021, February). Fault Analysis and Debugging of Microservice Systems: Industrial Survey, Benchmark System, and Empirical Study. *IEEE Transactions on Software Engineering, 47*, 243–260. doi:10.1109/TSE.2018.2887384

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.

Zipkin. (2022, 07 18). Retrieved from https://zipkin.io/