TRANSACT

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

D27 (D2.2)

System Test Platforms for Simulation based design and assessment of SCDCPS

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

ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership

Document Information

Project	TRANSACT
Grant Agreement No.	101007260
Work Package No.	WP2
Task No.	T2.2
Deliverable No.	D27
Deliverable No. in WP	D2.2
Deliverable Title	System Test Platforms for Simulation-based design and assessment of SCDCPS
Nature	Report
Dissemination Level	Public
Document Version	V1.0
Date	13/12/2023
Contact	Astrid Rakow
Organization	DLR
Phone	+49 441 770507 257
E-Mail	astrid.rakow@dlr.de

Authors Table

Name	Company	E-Mail
Sebastian Vander Maelen	DLR	Sebastian.vander.maelen@dlr.de
Eike Möhlmann	DLR	Eike.moehlmann@dlr.de
Astrid Rakow	DLR	Astrid.Rakow@dlr.de
Joan Josep Valls	ITI	Jvalls@iti.es
Robert Hofsink	PMS	Robert.hofsink@philips.com
Abel Gómez	UOC	agomezlla@uoc.edu
Iván David Alfonso	UOC	idalfonso@uoc.edu
Miguel García-Gordillo	ITI	Miguelgarcia@iti.es
Wolfram Ratzke	AVL	Wolfram.Ratzke@avl.com
Anas Aamoum	FLEET	anas.aamoum@fleetonomy.ai
Juhani Latvakoski	VTT	juhani.latvakoski@vtt.fi
Pertti Peussa	VTT	pertti.peussa@vtt.fi
Carlo Alberto Boano	TUG	cboano@tugraz.at
Adam Bachorek	IESE	adam.bachorek@iese.fraunhofer.de
Arnold Akkermann	DLR	arnold.akkermann@dlr.de
Jeanneth Nodland	NVT	jeanneth.nodland@navtor.com

Reviewers Table

Version	Date	Reviewer
V0.2	02/11/2023	Astrid Rakow(DLR), Eike Möhlmann(DLR)
V0.4	20/11/2023	Jeroen Voeten(TUE), Adam Bachorek(IESE), Mateusz Groth(GUT)

Change History

Version	Date	Reason for Change	Affected pages
V0.1	27/02/2023	First draft of the table of content	All
V0.2	02/11/2023	First Full Draft	All
V0.3	09/11/2023	Internally reviewed version	All
V0.4	20/11/2023	Version for Review	All
V0.5	01/12/2023	Commented Version	All
V0.6	08/12/2023	Submitted Version	All

Contents

List of Figures

Glossary

Term	Definition
Allocation (*aka* task allocation)	Task allocation refers to the decision of task placement and scheduling associated with the resource management.
Application migration	Application migration is the process of moving a software application from one computing environment to another. You might, for instance, migrate an application from one data centre to another, from an on-premises server to a cloud provider's environment, or from the public cloud to a private cloud environment.
Architecture	The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.
Architecture framework	Conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders.
Cloud service migration	[Cloud] service migration is a concept used in cloud computing implementation models that ensures that an individual or organization can easily shift between different cloud vendors without encountering implementation, integration, compatibility and interoperability issues.
Component	One of the parts that make up a system.
Computing platform	A computing platform is the environment in which a piece of software is executed. It may be the hardware or the operating system (OS), even a web browser and associated application programming interfaces, or other underlying software, as long as the program code is executed with it. Computing platforms have different abstraction levels, including a computer architecture, an OS, process containers, or runtime libraries. A computing platform is the stage on which computer programs can run.
Concept	An abstraction; a general idea inferred or derived from specific instances.
Cyber-Physical System	Digital system that semi-automatically interacts with its physical environment as integral part of its functionality. It integrates computation with physical processes, where system properties are determined by both cyber and physical parts.
Device	Physical entity embedded inside, or attached to, another physical entity in its vicinity, with capabilities to convey digital information from or to that physical entity.
Digital twin	A digital twin is a digital model of an actual physical system, which has a "live" connection (digital thread) with the physical system, so that it represents its actual status, and is used to derive a higher-level representation of the system's status and performance.

Edge Computing	Edge computing is a new architectural paradigm in which the resources of an edge server are placed at the edge of the Internet, in close proximity to cyber-physical systems, mobile devices, sensors and IoT endpoints.
Framework	A framework is an abstraction in which "engineering bricks" providing generic functionality can be selectively changed by system engineers, thus providing application-specific solutions. It provides a standard way to build and deploy CPS.
Industry 4.0	Industry 4.0 is a name given to the current trend of automation and data exchange in manufacturing technologies. It includes cyber-physical systems, the Internet of Things, cloud computing and cognitive computing. Industry 4.0 is commonly referred to as the fourth industrial revolution.
Mechanism	An established process by which something takes place or is brought about.
Method	A method consists of systematic steps for performing a task, in other words, it defines the "how" of each task.
Methodology	A collection of related formalisms, techniques, processes, methods, and tools. A methodology is essentially a "recipe" and can be thought of as the application of related processes, methods, and tools to a class of problems that all have something in common.
Middleware	Middleware is computer software that provides services to software applications beyond those available from the operating system. It can be described as "software glue". Middleware makes it easier for software developers to implement communication and input/output, so they can focus on the specific purpose of their application.
Mission-critical system	A mission-critical system is a system that is essential to the survival of a business or organization. When a mission-critical system fails or is interrupted, business operations are significantly impacted.
Mixed-criticality system	A system containing computer hardware and software that can execute several applications of different criticality, such as safety-critical and non-safety-critical, or of different Safety Integrity Level (SIL). Different criticality applications are engineered to different levels of assurance, with high criticality applications being the costliest to design and verify.
Offloading (*aka* computation or task offloading)	Computation offloading is the transfer of resource intensive computational tasks to a separate processor, such as a hardware accelerator, or an external platform, such as a cluster, grid, or a cloud. Offloading computing to an external platform over a network can provide computing power and overcome hardware limitations of a device, such as limited computational power, storage, and energy.
Orchestration	Type of composition where one particular element is used by the composition to oversee and direct the other elements.

	Note: The element that directs an orchestration is not part of the orchestration.
Partitioning (*aka* application or code partitioning)	Divides the application code into several parts that will be executed on different platforms, I.e., mobile devices, cloudlets, or the cloud.
Platform	A collection of interoperable system "engineering bricks" that can be used to set up a system engineering environment in a company. A technology or engineering brick can be: a software tool/product, a software component to build a software tool/product, a system engineering methodology, an interface, a standard, or means for establishing interoperability that is needed for the efficient development of safety-critical embedded systems.
Process	A process is a logical sequence of tasks performed to achieve a particular objective. A process defines "what" is to be done, without specifying "how" each task is performed.
Reference Architecture	A Reference Architecture (RA) is an architectural design pattern that indicates how an abstract set of mechanisms and relationships realizes a predetermined set of requirements. It captures the essence of the architecture of a collection of systems. The main purpose of a Reference Architecture is to provide guidance for the development of architectures. One or more reference architectures may be derived from a common reference model, to address different purposes/usages to which the Reference Model may be targeted.
Reference Model	A reference model is an abstract framework for understanding significant relationships among the entities of some environment. It enables the development of specific reference or concrete architectures using consistent standards or specifications supporting that environment. A reference model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or other concrete details. A reference model may be used as a basis for education and explaining standards to non- specialists.
Safety-critical CPS	A safety-critical CPS is a cyber-physical system where the failure or malfunction may result in one (or more) of the following outcomes: death or serious injury to people, loss or severe damage to equipment/property, environmental harm.
Safety-critical system	A system whose failure or malfunction may result in one (or more) of the following outcomes: death or serious injury to people, loss or severe damage to equipment/property, environmental harm.
Service	Services are the mechanism by which needs and capabilities are brought together.
Service migration	Dynamically moving (migrating) the service (or task) from one processing element to another at runtime.

Software component	An independent software unit that communicates with the surrounding system through explicitly specified interfaces.
Software framework	In computer programming, a software framework is an abstraction in which software providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific software. It provides a standard way to build and deploy applications and is a universal, reusable software environment that provides a particular functionality as part of a larger software platform to facilitate development of software applications, products and solutions. Software frameworks may include support programs, compilers, code libraries, tool sets, and application programming interfaces (APIs) that bring together all the different components to enable development of a project or system.
Solution	A means of solving a problem or dealing with a difficult situation.
System	A combination of interacting elements organized to achieve one or more stated purposes.
System component	A system architectural element.
Technique	Technical and managerial procedure that aids in the evaluation and improvement of the [system] development process.
Tool	A tool is an instrument that, when applied to a particular method, can enhance the efficiency of the task; provided it is applied properly and by somebody with proper skills and training.

Table 1: Terms and definitions

Term	Definition
AI	Artificial Intelligence
BDaaS	Big Data Analytics as a Service
CPA	Composite Performance Analysis
CPS	Cyber-Physical System
ECU	Electronic Control Unit
IoT	Internet of Things
MAPE	Monitor Analyse Plan Execute
MBSE	Model-based System Engineering
MDP	Markov Decision Process

ML	Machine Learning
PIM	Platform-Independent Metamodel
POOSL	Parallel Object-Oriented Specification Language
QoS	Quality of Service
RA	Reference Architecture
RTA	Response Time Analysis
SLA	Service Level Agreement
SLI	Service Level Indicator
SLO	Service Level Objective
SoA	State of the Art
SCDCPS	Safety-Critical Distributed Cyber-Physical System
WWTP	Wastewater Treatment Plant

Table 2: Abbreviations and definitions

1 Introduction

This report is a result of Task T2.2 of the TRANSACT project. The overarching goal of the TRANSACT project is to develop a universal, distributed solution architecture for the transformation of safety-critical cyber-physical systems, from localized standalone systems into safe and secure distributed solutions leveraging edge and cloud computing. To this end, four global project objectives have been defined, namely, “Leverage cloud technologies for safety-critical CPS” (Objective 1), “Enable continuous development and updates of applications and services” (Objective 2), ”Ensure safety and timing performance from a user perspective” (Objective 3) and “Guarantee security and privacy when using tools/data in cloud” (Objective 4). The work has been structured in six work packages. Work package WP2 is mainly concerned with “Distributed solution architectures and platforms for safety-critical CPS”**.** The goal of TRANSACT’s Task 2.2 is to develop simulation-based methods for the design and assessment of safety-critical distributed CPS solutions.

Task T2.2 targets the following list of four objectives:

- Obj. A: Specifications for services in distributed architectures
- Obj. B: Simulation of distributed safety-critical CPS solutions (device plus edge and cloud services) for a systematic design and early trade-off analysis (model-based design and exploration)
- Obj. C: Verification methods as a means of assurance and trusted interoperability between a wide range of safety critical cyber-physical systems of systems.
- Obj. D: Transition strategies for local CPS towards distributed safety-critical CPS solutions involving validation methods for evaluating alternative migration approaches.

The focus of Task T2.2 is on the objectives Obj. A and Obj. B. T2.2 and, hence, contributes mainly to the Objective 2 of the TRANSACT project, which focuses on enabling a continuous development and updates of applications and services in the device-edge-cloud continuum.

By Obj. C it also contributes to the project TRANSACT Objective 4 to guarantee security and privacy in the device-edge-cloud continuum.

In addition, Task T2.2 also contributes to Objective 1 of the TRANSACT project, which focuses on leveraging edge and cloud technologies for safety critical CPSs by its contribution to Obj. D.

1.1 Structure of the deliverable

In the rest of this section, we first describe the role of this deliverable (Sect. 1.2) and its relation with other deliverables of the TRANSACT project (Sect. 1.4).In Sect. 1.3 we give an overview of the activities of T2.2 and how they relate to the other activities undertaken in TRANSACT. In Sect. 1.4 we discuss the relation to other deliverables. In particular, we discuss the relation of the results of T2.2 to (i) Task T3.3 on solution for end-to-end safety and performance (D16(D3.3)) and to (ii) Task T3.4 on solutions for end-to-end security and privacy.

Section 2 describes the application context of the tools & methods that have been developed in T.2.2. To this end, the TRANSACT reference architecture (cf. D7(D2.1)) is described that is used within the different use cases and the use cases are briefly summarized.

Section 3 gives on overview of the general validation approach along the transition of a SCPS to a SCDCPS.

Section 4 presents the developed tools & methods in detail, each in its dedicated subsection.

We conclude in Section 5. In the appendix you will find a tabular overview of the tools and methods developed in T2.2.

1.2 Role of this deliverable

The main purpose of this deliverable is to present selected methods for simulation-based design and assessment of SCDCPS. To summarize the main tasks of this deliverable:

- We present the developed TRANSACT methods for simulation-based design and assessment of SCDCPS.
- We show how the developed methods are used within the design and validation process of SCDCPS.

  Since design and validation are deeply intertwined, we first discuss our reference validation process and then define a generic TRANSACT design process, highlighting the links to validation and assessment activities. We localize the methods of Task 2.2 within these validation and design processes.
- The deliverable moreover describes the motivation and importance of the selected methods by relating the properties targeted by the simulation methods with the system requirements that have jointly been determined by D5(D1.1) and D6(D1.2).
- The methods' role within the overall TRANSACT transformation is illustrated by showing which methods of T2.2 are applicable to which solutions of D16(D3.3) and D17(D3.4). We moreover explain their relation to the TRANSACT generic architecture of D7(D2.1).

1.3 Objectives and Topics of Task T.2.2

In this section, we show how the methods and tools of T2.2 contribute to all four objectives (Obj. A-D) and therefore to the Objectives 1, 2 and 4 of the TRANSACT project. To this end, we identify the necessary validation properties of trustworthy SCDCPS based on the technical requirements of the TRANSACT architecture as identified in D6(D1.2) and show how they are covered by the methods and tools developed in Task T2.2. Based on the validation properties, we derive four central topics that are particularly important for our work on the four objectives (Sect. 1.3.2).

1.3.1 Mapping between technical system requirements and validation properties

Task 1.2 of deliverable D6(D1.2) derives a list of TRANSACT System requirements from the user requirements that were initially collected within Task 1.1 (cf. D5(D1.1). As the goal of T2.2 is the development of simulation-based methods for the design and assessment of SCDCPS, we analysed which aspects of a trustworthy system were referred to by the TRANSACT System requirements that represent the generalized user requirements. This mapping of requirements to the aspects of trustworthiness guides us in determining the particular validation and assessment needs of the TRANSACT transition process. With other words, the transition process must create a trustworthy system and the methods of T2.2 assesses qualities of SCDCPS by means of simulation that constitute a trustworthy system and the methods focus on those aspects that need special attention as derived from the TRANSACT System requirements.

Next, we briefly introduce the notion of trustworthiness and then map the TRANSACT system requirements to the validation aspects that a trustworthy system must satisfy.

Literally "trustworthy" means that something is worthy to be trusted. Trustworthiness is estimated based on evidence or observations that assure that the system will perform as expected (Cho et al. 2016). The specific list of attributes that constitute trustworthiness varies in the literature. One reason for this is that different classes (Avizienis et al. 2004) of systems have been considered over time.

Avizienis et al. define dependability as the same concept as trustworthiness. It is based on functionality, performance, and dependability, coupled with security and cost. Dependability and security are affected by usability, manageability. Adaptability and security are based on availability, integrity, and confidentiality. In contrast, according to Hasselbring and Reussner (Hasselbring und Reussner 2006) the main attributes of software trustworthiness are correctness, safety, quality of service (availability, reliability, performance), security (availability, integrity, confidentiality), and privacy.

For structuring the validation aspects of T2.2 in TRANSACT, we used the trustworthiness attributes of the more recent paper (H. J. Putzer, H. Rues, J. Koch) by Putzer et al. on trustworthy AI-based systems. According to (H. J. Putzer, H. Rues, J. Koch), a trustworthy system must be reliable, available, maintainable, safe, secure, it must ensure privacy and implement ethical aspects and it must be useable and robust, as is illustrated in Figure 1.

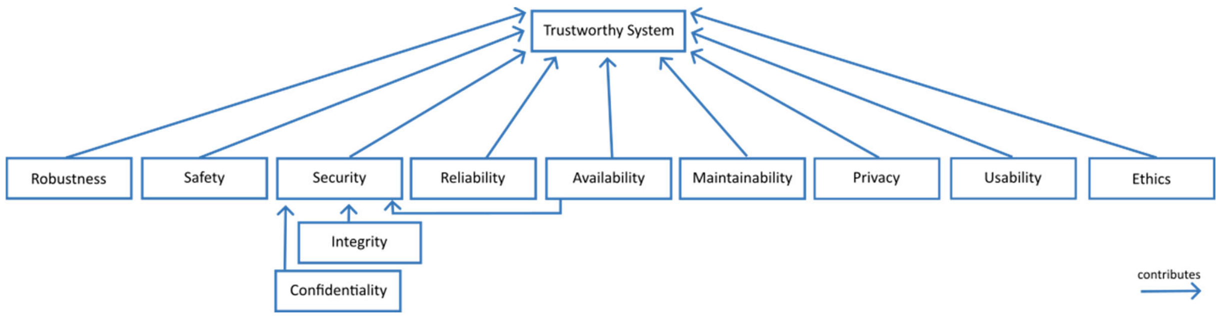


Figure 1: Aspects of Trustworthiness

To avoid confusion, we briefly sketch our interpretation of these attributes, mainly following (Avizienis et al. 2004). We use “robustness” for “continuity of correct service”, “safety” for “freedom from danger, risk, or injury” and it includes system safety, functional safety and safety of intended functionality (SOTIF), “security” for “protection from hostile forces”. Security is based on “integrity”, i.e. the absence of improper system alterations, and “confidentiality”, the absence of unauthorized disclosure of information. We use “reliability” for “the continuity of correct service”, while “availability “ refers to “the readiness of the system for correct service”. ”Maintainability” means “the ability to undergo modifications and repair”, “usability” is the result of effectiveness, efficiency and the qualities “easy to learn” and “fun to use”. “Ethics” includes legal, social and societal concerns. Moreover, “dependability” results from “maintainability”, “safety”, “reliability”, “availability” and “security”.
The simulation-based methods of T2.2 must validate/assess these different aspects to ensure that the TRANSACT transition process leads to a trustworthy system. Based on the end user requirements from Task T1.1 (cf. D5(D1.1)), deliverable D6(D1.2) of T1.2 describes the functional, non-functional and technical requirements and classify the abstracted Technical Requirements for the TRANSACT System (TSR) into four groups: security relevant requirements, safety relevant requirements, additional safety & performance requirements, and requirements for the implementation of an AI.

For Task T2.2, we have adopted the approach of classifying the TSRs, but with the aim of deriving from the TSRs the need for a simulative assessment of the system under development. Therefore, we classified the TSRs with respect to the aspects of trustworthiness. While some TSRs are directly mapped to these aspects, other TSR were grouped into classes that represent key instrumental properties -- they must be established to achieve other aspects of a trustworthy system. These additional properties relate to the dealing with faults, as faults within the system can jeopardize trustworthiness. Table 3 shows an excerpt of the mapping from the requirements of D6(D1.2) to the validation aspects. As a result of this process, we derived the validation aspects as illustrated in Figure 2.

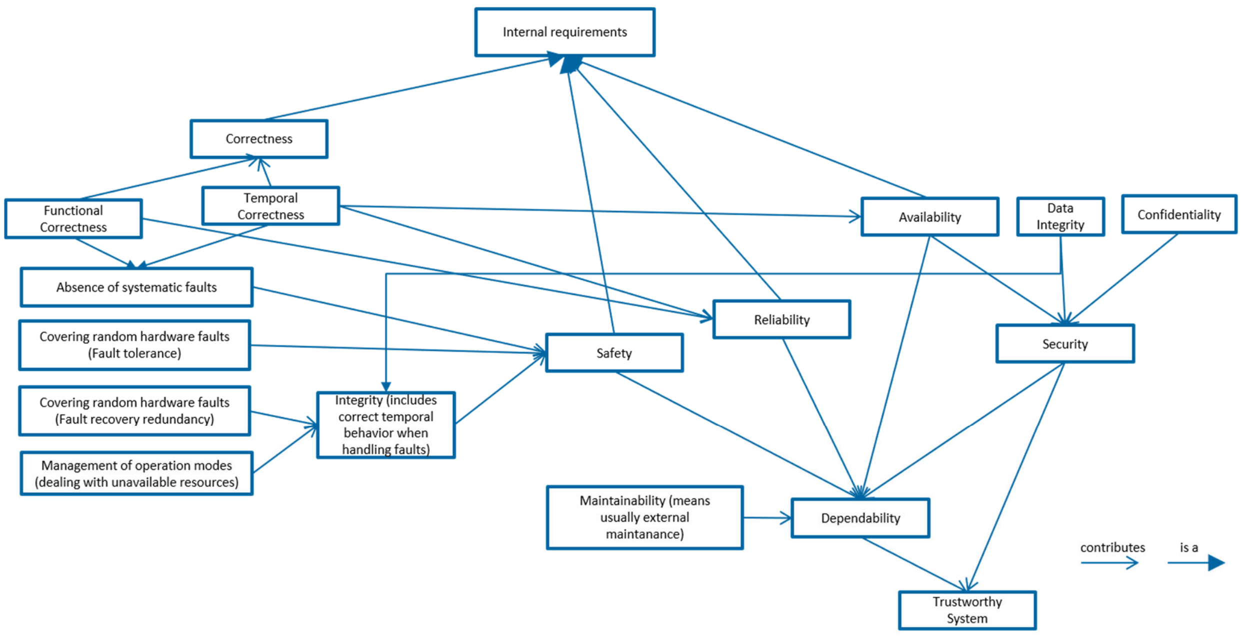
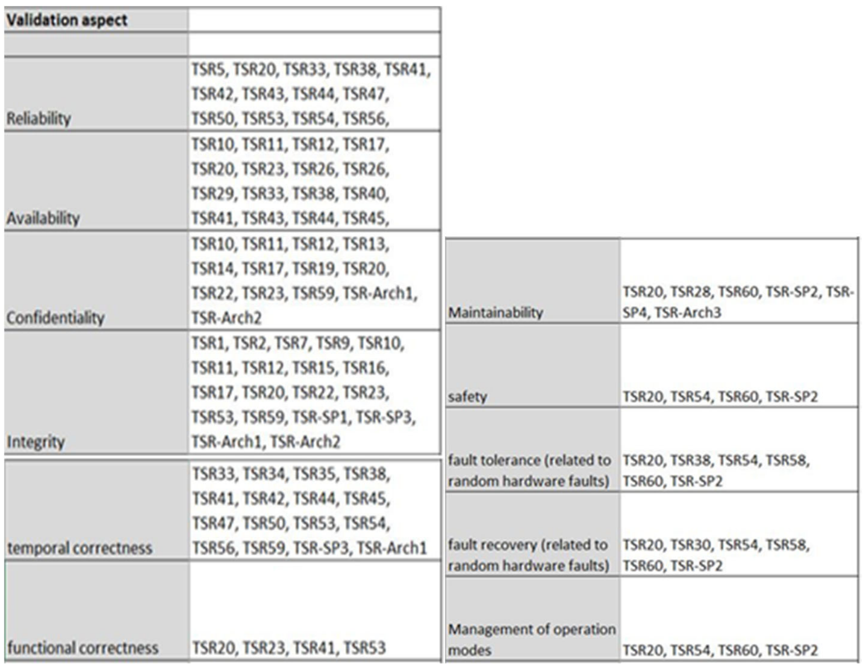

Internal requirements
Correctness
Functional Correctness
Temporal Correctness
Absence of systematic faults
Covering random hardware faults (Fault tolerance)
Covering random hardware faults (Fault recovery redundancy)
Management of operation modes (dealing with unavailable resources)
Integrity (includes correct temporal behavior when handling faults)
Safety
Reliability
Availability
Data Integrity
Confidentiality
Security
Maintainability (means usually external maintanance)
Dependability
Trustworthy System
contributes
is a

Figure 2: Aspects of trustworthiness and validation properties derived from the TSRs



Validation aspect	
Reliability	TSR5, TSR20, TSR33, TSR38, TSR41, TSR42, TSR43, TSR44, TSR47, TSR50, TSR53, TSR54, TSR56,
Availability	TSR10, TSR11, TSR12, TSR17, TSR20, TSR23, TSR26, TSR26, TSR29, TSR33, TSR38, TSR40, TSR41, TSR43, TSR44, TSR45,
Confidentiality	TSR10, TSR11, TSR12, TSR13, TSR14, TSR17, TSR19, TSR20, TSR22, TSR23, TSR59, TSR-Arch1, TSR-Arch2
Integrity	TSR1, TSR2, TSR7, TSR9, TSR10, TSR11, TSR12, TSR15, TSR16, TSR17, TSR20, TSR22, TSR23, TSR53, TSR59, TSR-SP1, TSR-SP3, TSR-Arch1, TSR-Arch2
temporal correctness	TSR33, TSR34, TSR35, TSR38, TSR41, TSR42, TSR44, TSR45, TSR47, TSR50, TSR53, TSR54, TSR56, TSR59, TSR-SP3, TSR-Arch1
functional correctness	TSR20, TSR23, TSR41, TSR53
Maintainability	TSR20, TSR28, TSR60, TSR-SP2, TSR-SP4, TSR-Arch3
safety	TSR20, TSR54, TSR60, TSR-SP2
fault tolerance (related to random hardware faults)	TSR20, TSR38, TSR54, TSR58, TSR60, TSR-SP2
fault recovery (related to random hardware faults)	TSR20, TSR30, TSR54, TSR58, TSR60, TSR-SP2
Management of operation modes	TSR20, TSR54, TSR60, TSR-SP2

Table 3: Mapping of Requirements from D1.2(D6) to validation aspects

As a result of T2.2, several simulation and specification methods have been extended/developed as systematic approaches to analysis, verification and validation. All these methods and tools support the design of a trustworthy system – albeit focusing on different aspects. We briefly list the methods & tools pointing out which trustworthiness aspect is targeted and providing a link to the respective section that describes the contribution in detail. An overview of the relation between the methods & tools and the covered trustworthiness aspect is given in Figure 3. A tool/method is represented by the number of its respective subsection within Sect. 4 *(1=”V&V Backend Architecture”, 2=”Risk Storming”, 3=”V&V Concept for Maritime[…], 4=” A2K”; 5=”Validation of Image Guided Therapy”, 6=”Hybrid Simulation”, 7=”Co- and Parallel Simulation” and 8=”Model Transformations”)*.

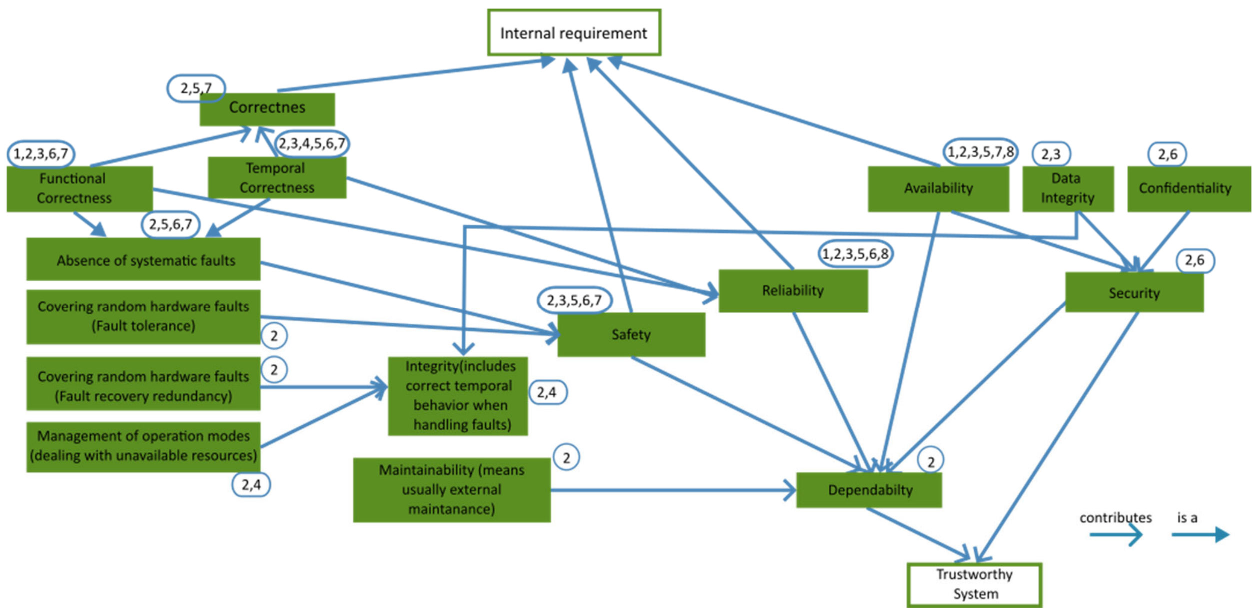

Internal requirement
2,5,7
Correctnes
1,2,3,6,7
Functional Correctness
2,3,4,5,6,7
Temporal Correctness
2,5,6,7
Absence of systematic faults
Covering random hardware faults (Fault tolerance)
2
Covering random hardware faults (Fault recovery redundancy)
2
Management of operation modes (dealing with unavailable resources)
2,4
Integrity(includes correct temporal behavior when handling faults)
2,4
Maintainability (means usually external maintanance)
2
2,3,5,6,7
Safety
1,2,3,5,6,8
Reliability
1,2,3,5,7,8
Availability
2,3
Data Integrity
2,6
Confidentiality
2,6
Security
2
Dependabilty
Trustworthy System
contributes
is a

Figure 3: Coverage of the aspects of trustworthiness by the results of T2.2.

An approach to validation and verification of a backend architecture has been developed (cf. Sect. 4.1) that focuses on testing how the system reacts when scaling of the number of clients by virtual vehicle/digital twin. The approach targets functional correctness, reliability and availability.

By providing a visual and collaborative risk identification technique (cf. Sect. 4.2), the risk storming approach based on the C4-Model[1] identifies the general risks in the software design. This approach contributes to all aspects of trustworthiness.

A simulation-based validation and verification framework (cf. Sect. 4.3) for the maritime domain based on the HAGGIS (Hybrid Architecture for Granularly, Generic and Interoperable Simulations) simulator checks the quality of routes and navigational safety, and deals with the risks introduced by machine learning using AIS data within the advisory system (MIL&SIL simulation). The approach targets functional correctness, temporal correctness, safety, reliability, availability, and data integrity.

[1] https://c4model.com/

Art2kitekt (Sect. 4.4) allows for the evaluation of different distributed applications in the device-edge-cloud continuum during the steady state, and the evaluation of the dynamic behavior of multimode applications during the transient state. Art2kitekt thereby targets the temporal correctness, the integrity of the system and the management of operation modes.

The validation of image-guided therapy (Sect. 4.5) focuses on the evolution of performance models. It targets correctness via temporal correctness and the absence of systematic faults, as well as safety, reliability and availability.

The hybrid simulation-based validation of remote driving (Sect. 4.6) evaluates the situation awareness of vehicles, examines how to facilitate and evaluate traceability. It targets security, privacy and trust in a collaborative system. The overall approach supports functional and temporal correctness, the absence of systematic faults, safety, reliability, security and confidentiality.

Co-simulation and parallel simulation (Sect. 4.7) using FERAL facilitates the evaluation of the cloud-featured battery management system with a focus on communication (CAN/cellular network). The work supports correctness, temporal correctness, absence of systematic faults, safety and availability.

Model transformations (Sect. 4.8) can be used to generate different deployments from different models. By using and replaying historical data at different paces, these model transformations can be leveraged to simulate a synthetic workload, testing whether the deployments can keep up with it or not. This contributes to validate the reliability and availability non-functional requirements.

1.3.2 Relationship between developed methods & tools and the objectives

After the validation aspects (cf. Figure 2) have been derived from the TSRs, the four key topics

- Simulation Tools and Platforms
- Simulation based Design
- Simulation based Analysis, Verification and Validation
- Model based Analysis

were identified by the partners as the most important focal points for achieving T2.2's objectives. For the convenience of the reader, we present the list the objectives from p. 14 here:

- Obj. A: Specifications for services in distributed architectures
- Obj. B: Simulation of distributed safety-critical CPS solutions for systematic design and early trade-off analysis
- Obj. C: Verification methods as a means of assurance and trusted interoperability between a wide range of safety critical cyber-physical systems of systems.
- Obj. D: Transition strategies for local CPS towards distributed safety-critical CPS solutions involving validation methods for evaluating alternative migration approaches.

To support the design and enable a trade-off analysis (Obj. B) the partners have developed "simulation tools and platforms" as well as "model-based analysis". It is explored how services in a distributed architecture can be specified (Obj. A). The topic of "simulation-based analysis, verification and validation" is a direct means to achieving Obj. C when the interaction of system components is examined. The simulative verification and assessment steps accompany the system development process. They must, hence, be aligned with the TRANSACT transition process. To this end, a general transition strategy has been developed (Obj. D).

Figure 4 shows which tool/method is the result of the work on which core topic. Again a tool/method is represented by the number of its respective subsection subsection within Sect. 4 *(1="V&V Backend Architecture", 2="Risk Storming", 3="V&V Concept for Maritime[…], 4=" A2K"; 5="Validation of Image Guided Therapy", 6="Hybrid Simulation", 7="Co- and Parallel Simulation" and 8="Model Transformations")*. Figure 5**Error! Reference source not found.** shows which partner contributes to which topic by either contributing directly a tool/method or by contributing expertise within TRANSACT.

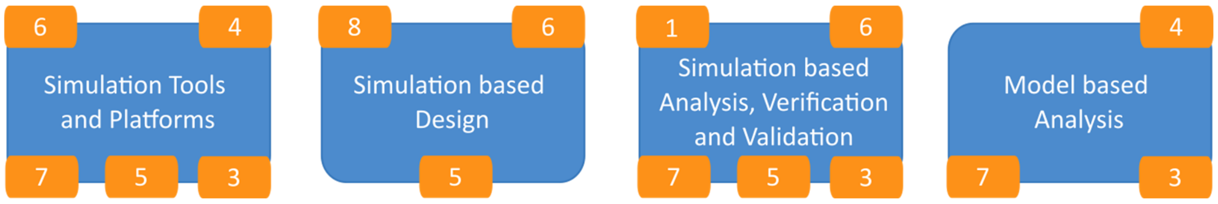

6
4
Simulation Tools and Platforms
7
5
3
8
6
Simulation based Design
5
1
6
Simulation based Analysis, Verification and Validation
7
5
3
4
Model based Analysis
7
3

Figure 4: Core topics and the respective methods & tools

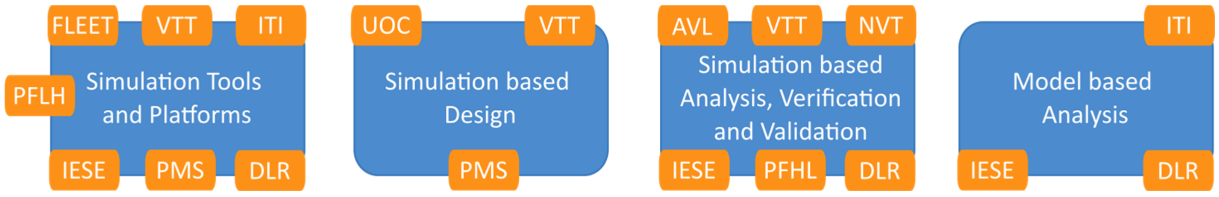


Figure 5: Core topics and contributing partner

1.4 Relationship to other TRANSACT deliverables

1.4.1 Mapping of developed methods/tools to the concepts developed in D3.2, D3.3 and D3.4

The methods & tools of T2.2 have been developed in the context of concepts/solutions created within TRANSACT to ensure a trustworthy SCDCPS. Some of the methods & tools developed in T2.2 are hence specifically aimed at the simulation/assessment of these realisations. In the following, we list the relations between the methods & tools of T2.2 with the concepts/solutions as described in other deliverables. The results of T2.2 relate to concepts/solutions of the deliverables D9(D3.2) D15(D3.3), and D16(3.4). The deliverables D9(D3.2) presents a selection of concepts for end-to-end security and privacy for distributed CPS solutions, D15(D3.3) presents solutions for end-to-end safety and performance for distributed CPS solutions while D16(D3.4) presents solutions for end-to-end security and privacy for distributed CPS solutions.

AVL's V&V of Backend Architecture (Sect. 4.1) allows to examine Solution "S3 Solutions for scalable applications" and "S6 Performance observability and monitoring" of D3.3 and "11.2 Hardware security: HSM to store keys on edge device" of D3.4.

DLR's simulation-based V&V concept (Sect. 4.3) executes a solution of "4.3 Solutions for safety and health monitoring, risk analysis and safety and security" and the simulation approach allows examining "S1 Mode change management on the device" and "S5 Solutions for ensuring data integrity".

Art2kitekt (Sect. 4.4) allows examining "S1 Mode change management on the device" and "S2 Mode change coordination" as well as "S6 Performance observability and monitoring" of D3.3.

Philips' validation of image-guided therapy (Sect. 4.5) allows via POOSL "S8 Simulation based performance analyses" of D3.3.

Hybrid simulation (Sect. 4.6) supports "6.2 Security, privacy and trust related solutions for remote driving of operation" of D3.4 and instantiates "Requirements for Security and privacy concepts for secure remote driving operation" of D3.2, Sect: 9.1.

1.4.2 Related Deliverables

This document relates to the following TRANSACT deliverables:

- The TRANSACT reference architecture for SCDCPS is presented in D26 (D2.1). The developed tools & methods of T2.2 are tailored to this architecture, as will be described in the following. The reference architecture will be briefly introduced in Sect. 2.1 for convenience of the reader. Each of the contributions in Sect. 4 will state what instantiation of the reference architecture they consider.
- The TRANSACT technical requirements (TSR) are presented in D6(D1.2) and have been used to define the focus of the activities in task T2.2.
- The deliverables D8(3.1) and D15(D3.3) present concepts/solutions for end-to-end safety and performance for distributed CPS solutions while D9(3.2) and D16(D3.4) present concepts/solutions for end-to-end security and privacy for distributed CPS solutions. The methods & tools of T2.2 have been developed within the context of these approaches and aim to simulate or assess their realisations, as described in more detail in Sect. 4.
- The tool & methods of T2.2 have been developed driven by the TRANSACT use cases. Sect. 2 gives an overview to make this document self-contained. More information on the use cases can be found in D5(D1.1).
- The methods & tools of T2.2 are for assessment and evaluation of a system during the system development process as described in TRANSACT transition guide D23(D1.3).

  The results obtained by this deliverable document D27(D2.2) will be used to validate the demonstrators in WP5. It is expected to be aligned with the use cases.

2 Use Cases and the reference architecture

In this section, the context is described in which the methods & tools of T2.2 have been developed and applied. This section aims to make this deliverable self-contained. It, hence, describes the use cases and the reference architecture only briefly. It explains the main challenges addressed and what instance of the TRANSACT reference architecture has been considered in the use cases.

Since all use cases developed a specific instantiation of the TRANSACT architecture, we briefly describe the generic TRANSACT reference architecture first. The concrete instantiations are described as part of the use cases. More information on the use cases can be found in D5(D1.1) and more information on the reference architecture can be found in D7(D2.1).

2.1 Reference Architecture

In the following, we present the TRANSACT architecture concept that has been developed within task T2.1 for distributed safety-critical CPS solutions. To this end, we briefly summarize Section 3 of Deliverable D7(D2.1).

The TRANSACT architecture concept (cf. Figure 6) distinguishes three tiers: the device tier, the edge tier, and the cloud tier. The CPS end devices are linked via the edge infrastructure to the cloud computing facilities, where the end device and the edge tier are connected via a reliable network, whereas the cloud tier has only access to the internet. TRANSACT defines the core components (shown in grey) for all three layers that allow functions to be offloaded from the end device to off-device tiers. Both off-device tiers, the edge and cloud tier, are safe and secure as well as multi-application, multi-device, possibly multi-tenant, scalable and interoperable.

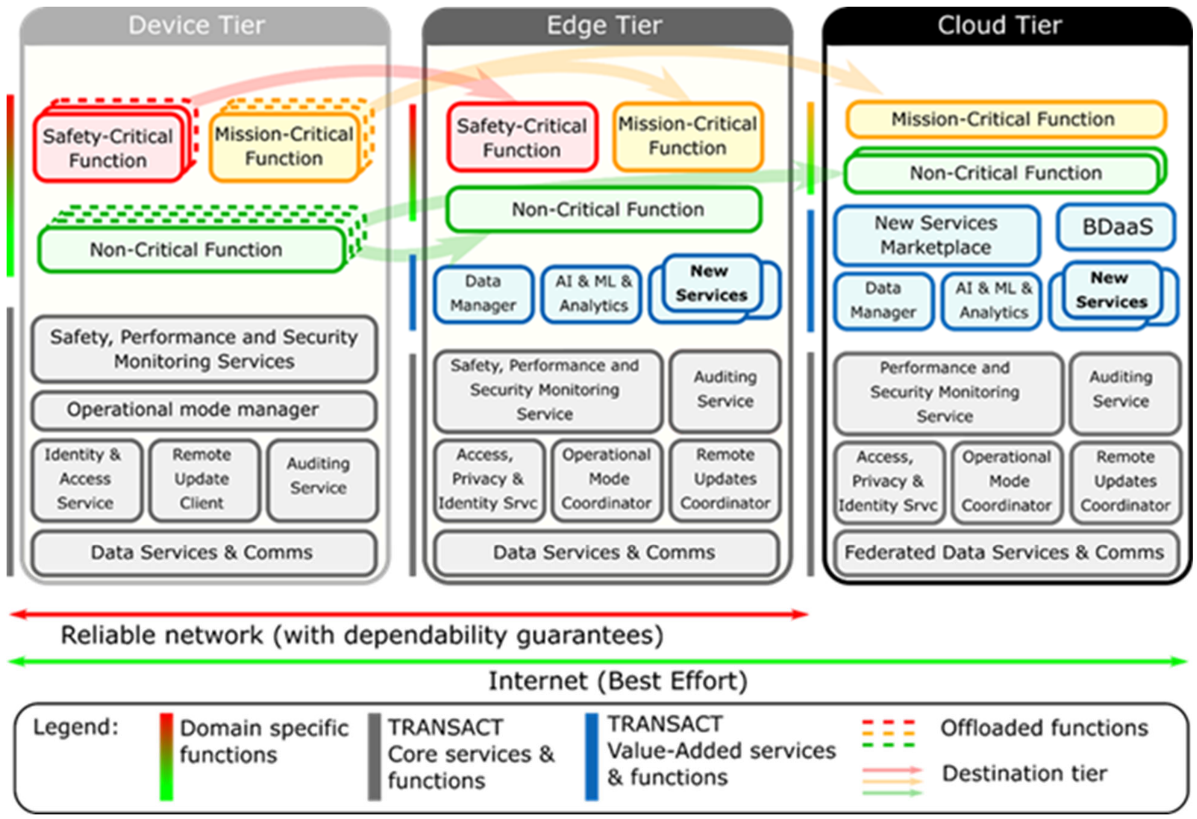

Device Tier
Edge Tier
Cloud Tier
Safety-Critical Function
Mission-Critical Function
Non-Critical Function
Safety, Performance and Security Monitoring Services
Operational mode manager
Identity & Access Service
Remote Update Client
Auditing Service
Data Services & Comms
Data Manager
AI & ML & Analytics
New Services
Safety, Performance and Security Monitoring Service
Access, Privacy & Identity Srvc
Operational Mode Coordinator
Remote Updates Coordinator
New Services Marketplace
BDaaS
Performance and Security Monitoring Service
Federated Data Services & Comms
Reliable network (with dependability guarantees)
Internet (Best Effort)
Legend:
Domain specific functions
TRANSACT Core services & functions
TRANSACT Value-Added services & functions
Offloaded functions
Destination tier

Figure 6 The TRANSACT architecture concept

The TRANSACT transformation approach distinguishes services according to their criticality: Non-critical services (depicted in green) and mission-critical services (shown in yellow) are offloaded to the edge tier and the cloud tier for execution. Then certain safety-critical functions are moved to the edge tier. The system capabilities can furthermore be enhanced by value-added functions (shown in blue).

The end-to-end safety and security of the CPS solutions are jointly realized by dedicated core services. *Safety, performance and security monitoring services* monitor, detect, and prevent (safety, security and performance) failures. They also measure the Service Level Indicators (SLIs). If the SLIs indicate that the Service Level Objectives (SLOs) are compromised or out of range, the *operational mode manager* on the device and the *operational mode coordinator* on the edge/cloud tier must decide at run-time on the appropriate operational mode. The mode management concept is a central means of the TRANSACT approach to ensure safety and reliability, we hence give more details in Sect. 2.1.1.

For updating the distributed CPS solution, the TRANSACT architecture provides the *remote update client* on the device and the *update coordinator* on the edge/cloud. To realize safe and secure remote updates of the device services these services work together across tiers. Any update activity is coordinated with the operational mode coordinator service to keep the system safe.

Security aspects are addressed by the *access, privacy and identity services*, which grant or deny access to the system resources based on the access policies. The *auditing services* collect information on accesses and system usage to detect security policy violations. The *(federated) data and comms services* assist in efficient and secured data handling, when the system is in transit and at rest.

Value-added services (shown in blue) extend the capabilities of the system. New value-added services can be provided not only by the system developer but also by 3rd party providers through the *new services marketplace*. Moreover, there are *data manager services* for the storage, protection and accessing data in the off-device tiers. The *Big Data as a Service (BDaaS) component* provides advanced analyses of big data sets to other services. The *AI & ML & analytics services* give insights into the collected data. This information is used to improve users' tasks or to optimize a company's operational performance and costs.

2.1.1 Dynamics of the TRANSACT architecture & Mode Management

The concept of operational mode management is a central means in TRANSACT to ensure the security and reliability of the CPS solution at rest or during an update. In the following, we illustrate the dynamics of the TRANSACT architecture concept with a mode management scenario shown in Figure 7. For more information on operational mode management, see Deliverable D7(D2.1), Section 4.4.

A function at the cloud level triggers an event. For example, a safety monitoring service detects that an SLI has reached a threshold. The event causes the Operational Mode Coordinator (OMC) at the cloud tier to analyse the situation. The OMC deduces that the mode A must be changed to mode B. In our scenario, the event indicates that the SLO cannot be guaranteed for much longer. So, the OMC requests a global mode change from the edge tier OMC. The edge tier OMC analyses the request of changing to mode B and concludes that the device needs to change locally from mode 1 to mode 2. So, it requests this local mode change from the Operational Mode Manager (OMM) of the device tier. In our scenario, the request can be fulfilled, and the local mode change takes place. This is reported back to the cloud tier via the edge tier. Only when the OMM of the cloud tier receives notification from the edge tier that the mode change has been completed, the global mode is changed to mode B on the cloud tier.

At some later time, the function on the device tier triggers an event and OMM infers that a local mode change is required. This is analogous to 1 and 2 in Figure 7, but then the local mode 2 is changed directly to mode 3. This forced change is communicated by the OMM to the OMC on the edge tier. The OMC in turn reports to the OMC of the cloud tier that the mode change from mode 2 to mode 3 has been enforced on the device. The OMC of the cloud tier then adjusts the global mode and notifies the edge tier.

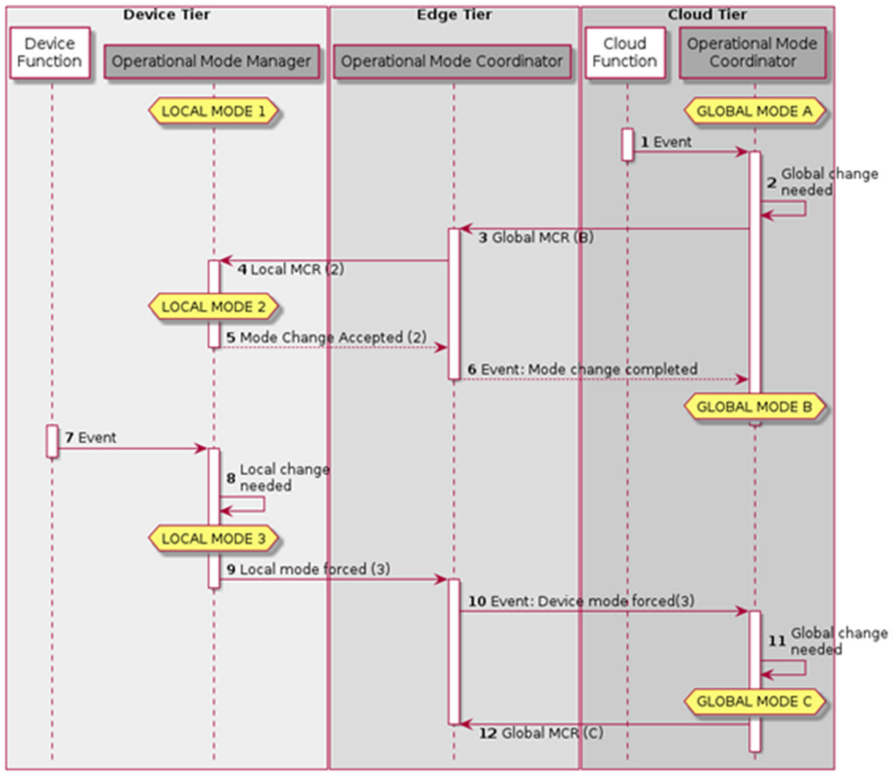


Figure 7: Mode Changing Scenarios Across the Tiers

2.1.2 Implications for Methods & Tools for the Simulation & Assessment

As outlined in D23(D1.3) the three focal areas for the transition process are business, solution architecture, and organization, and there are key cross-cutting aspects -- safety, performance, security and privacy, and regulatory and certification -- that impact considerably the transition steps. These require special attention to ensure the success of such a transition. The architecture introduces several core services that are deployed across all tiers to ensure safety, performance and security of the new solution and prevent safety, security and performance failures (cf. p. 11 of D23(D1.3)).

As outlined in D23(D1.3), during the development the cross-cutting concerns are examined via dedicated views, so that the system is designed to be safe, secure and respect the performance requirements. To this end, model driven design is recommended. Additionally, the importance of observability by design is stressed. The importance and impact of the cross-cutting concerns is key to a trustworthy system. As illustrated in D23(D1.3), p. 16, by the meaning of performance constraints. Safety-critical functions often have hard real-time constraints. Their failures can lead to severe safety-critical failures. Mission-critical functions tend to have soft real-time constraints. When missed, they can degrade the system's quality of service.

The system test platforms for simulation-based design and assessment target to support the front-loading of tests within the system development. As early as possible the properties of a design alternative should be assessed. E.g., the V&V backend architecture allows to assess scaling capabilities of the system under development early on by simulating the vehicle, i.e., virtualizing the edge and device so that vertical scaling is possible. The model transformations of Sect 4.8 allow to simulate an early system model within a simulation and monitoring environment that is created by a code generator from a CPS architecture model.

The TRANSACT process refers to a transition from an initial system to an evolved system within a process of continuous development and operation (DevOps). The approach to verification and validation hence exploits data and functionalities of the initial system, as it has been done, e.g., in the V&V concept for the maritime advisory system. Since only parts of the system are redesigned, hybrid simulation techniques for remote driving allow the assessment of the system within the concrete system context early on.

The high complexity of SCDCPS necessitates simulation approaches by which the system properties of interest can be examined in a preferably holistic evaluation environment based on authentic models with different abstraction levels if required, which is inherently taken into account by the co-simulation & parallel simulation approach.

2.2 Overview of Use Cases

The partner's methods and tools have evolved within the different use cases where they have been applied and evaluated. Figure 8 gives an overview of the use cases with the methods and tools of T2.2.

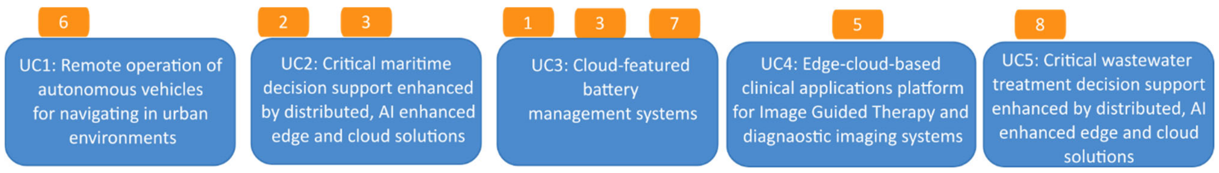

6
UC1: Remote operation of autonomous vehicles for navigating in urban environments
2
3
UC2: Critical maritime decision support enhanced by distributed, AI enhanced edge and cloud solutions
1
3
7
UC3: Cloud-featured battery management systems
5
UC4: Edge-cloud-based clinical applications platform for Image Guided Therapy and diagnaostic imaging systems
8
UC5: Critical wastewater treatment decision support enhanced by distributed, AI enhanced edge and cloud solutions

Figure 8: Use Cases with the methods and tools of T2.2

2.2.1 Use Case 1 - Remote Operations of Autonomous Vehicles

The remote operation of autonomous vehicles use case (UC1) will demonstrate how to manage the complex behaviour of a fleet of vehicles in an urban environment. This will be achieved through hybrid simulation combining, a remotely driven vehicle (HITL) and simulated vehicles (SITL) operating both, in a real and a simulated environment.

The main challenges are:

- Creating the digital twin of the environment
- Creating the digital twin of the vehicle
- Creating Simulations of the vehicle
- Feeding traffic info to the MQTT server
- Developing autonomous and remote-control modes for the vehicle
- Making a hybrid simulation that contains both, the real vehicle and simulated vehicles

In UC1 the TRANSACT architecture is instantiated as shown in Figure 9. The components are described in more detail via Table 4: Components in UC1.

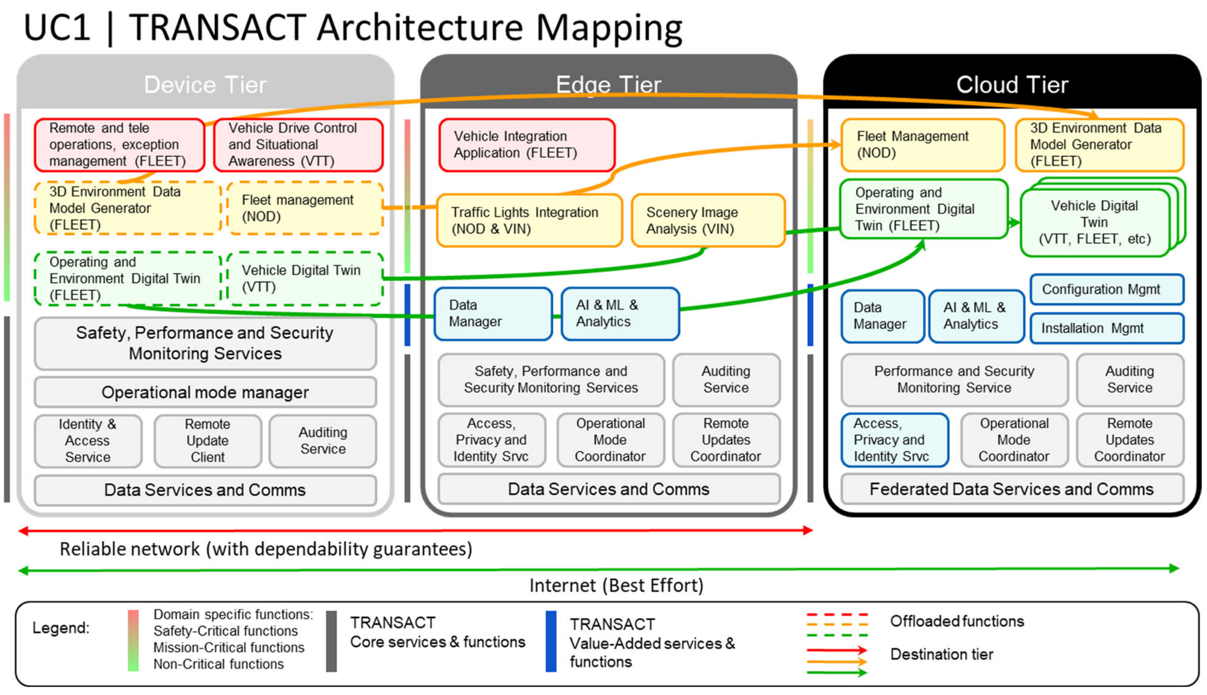

UC1 | TRANSACT Architecture Mapping
Device Tier
Edge Tier
Cloud Tier
Remote and tele operations, exception management (FLEET)
Vehicle Drive Control and Situational Awareness (VTT)
3D Environment Data Model Generator (FLEET)
Fleet management (NOD)
Operating and Environment Digital Twin (FLEET)
Vehicle Digital Twin (VTT)
Safety, Performance and Security Monitoring Services
Operational mode manager
Identity & Access Service
Remote Update Client
Auditing Service
Data Services and Comms
Vehicle Integration Application (FLEET)
Traffic Lights Integration (NOD & VIN)
Scenery Image Analysis (VIN)
Data Manager
AI & ML & Analytics
Safety, Performance and Security Monitoring Services
Auditing Service
Access, Privacy and Identity Srvc
Operational Mode Coordinator
Remote Updates Coordinator
Data Services and Comms
Fleet Management (NOD)
3D Environment Data Model Generator (FLEET)
Operating and Environment Digital Twin (FLEET)
Vehicle Digital Twin (VTT, FLEET, etc)
Data Manager
AI & ML & Analytics
Configuration Mgmt
Installation Mgmt
Performance and Security Monitoring Service
Auditing Service
Access, Privacy and Identity Srvc
Operational Mode Coordinator
Remote Updates Coordinator
Federated Data Services and Comms
Reliable network (with dependability guarantees)
Internet (Best Effort)
Legend:
Domain specific functions:
Safety-Critical functions
Mission-Critical functions
Non-Critical functions
TRANSACT Core services & functions
TRANSACT Value-Added services & functions
Offloaded functions
Destination tier

Figure 9: TRANSACT architecture and transition in UC1

Table 4: Components in UC1

Component	Description

3D Environment Data Model Generator	Simulation of the environment where the vehicles will be driven
Vehicle Drive Control and situational Awareness	An embedded component in the vehicle used for control, autonomous driving and status updates
Remote and tele operation, exception management	Software component for remote control of the vehicle. It is used also to raise up flags in case of errors
Operating and Environment Digital Twin	Software modelling of the environment where the vehicles are operating
Nodeon - Fleet management	A cloud app to provide UI for route design, real-time status of vehicles and geo-block design tool
Aune - VTT Demonstration Vehicle	Aune is a 6-person automated minibus. During remote driving, Aune is controlled with the FMA (see below).
CPS Trust monitor (VTT)	The component focuses on experimental digital trust and monitoring solutions needed in the case.
Vehicle Digital Twin	Software representation of the vehicle. It is used for simulation, testing and monitoring
Vehicle Integration Application	API between the fleet management application and the vehicle's digital twin
Traffic Lights Integration	Integration of data from traffic lights to the fleet management application
ViNotion Roadside Camera system	ViNotion Roadside Camera system (ViSense TrafficDynamics). Camera system gives ahead information about motorized traffic and vulnerable road users at the location, to which Aune is approaching.
FMA – Fleetonomy.ai**'s operating system for autonomous vehicle fleet**	Digital Twin of the vehicles and the area of operations, and user interfaces to the Mission Planner, the Remote Driver and the Ground Crew.

2.2.2 Use Case 2 - Maritime Monitoring and Decision Support System Supplemented by AI-enhanced Edge and Cloud Solutions

The maritime use case (UC2) will demonstrate advancements in safe and efficient maritime navigation, made possible by extending and enhancing the existing basic edge/cloud technologies in the NAVTOR e-Navigation Suite. This will be achieved by integrating advisory services, AI-based services, and data-analytics services into the device-edge-cloud continuum. The demonstrators will show a more integrated and connected architecture allowing for enhanced decision support and common situational awareness for three end users: the navigator on-board ships, the operator on the shore-based bridge, and the operator of the unmanned service vessel in harbour areas.

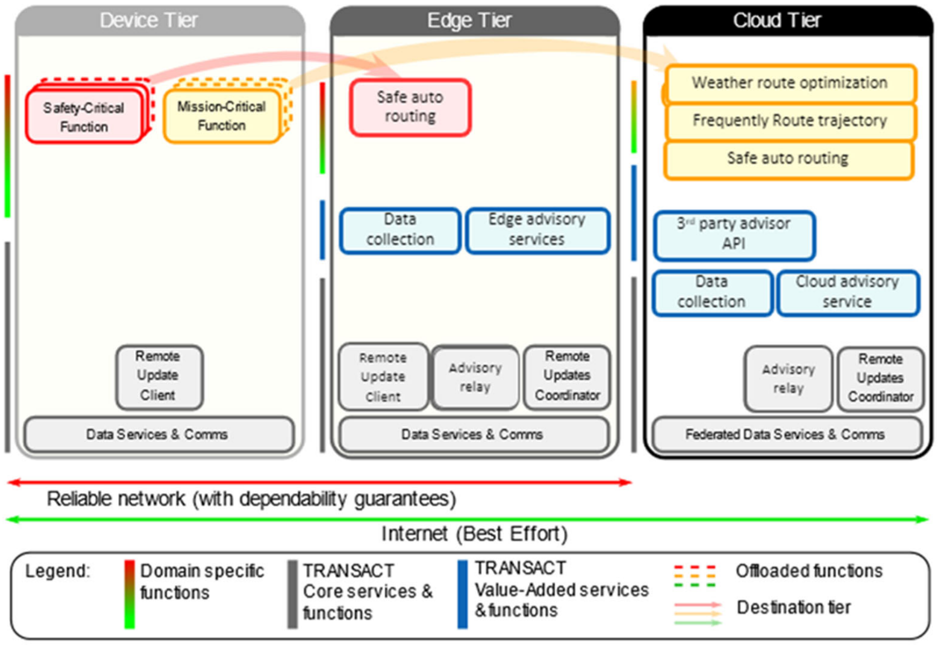


Figure 10: UC2 against TRANSACT reference mapping

UC2 needs to develop the advisory service framework allowing the request of advice to flow from device to cloud to be processed and back again to the device to be displayed, as well as to develop or integrate advisory services that process the advice based on available data. The necessary components to achieve this are listed here (ref. D5 (D1.1) Use case descriptions, end user requirements, SotA and KPI's)

Table 5: Components in UC2

Component	Description
External advisor interface	Develop APIs to interface with an external advisor
Functionality for Advice <u>response</u> relay in cloud	Extend functionality to receive and relay requests within NAVTOR platform
Functionality for Advice <u>request</u> relay in cloud	Extend functionality to receive and relay requests within NAVTOR platform
Cloud Advisory Services	Processing of requests/generating of suggestions for route optimization
Analytical Engines	Manually produce a baseline static decision model for routing advisory
Module to support data collection processes	Extend the current NAVTOR module that reads external data (such as AIS data), so that it can support additional data readers.
Functionality for Advice *request* relay on edge	Extend functionality to receive and relay requests from and to vessel devices
Functionality for Advice *response* relay on edge	Extend functionality to receive and relay response from and to vessel devices

Edge Advisory Services	Develop framework for advisory services on the edge; Develop an auto routing advisory service on the edge
Functionality for Advisory *request* on device	Develop/Extend GUI for requesting advice
Functionality for Advisory *handling* on device	Develop/Extend GUI(s) for visualisation and review of received advice

The simulation-based assessment within UC2 realize a quality check of the routes provided by the advisory service. The simulation moreover is concerned with checking navigational safety, also with respect to compliance with regulatory restrictions. To this end a depth check is provided by simulation-based solution. These checks also aim for dealing with the risks introduced by using AIS data for machine learning.

2.2.3 Use Case 3 – Cloud-Featured Battery Management System

Today’s vehicles are local and isolated data silos. Use Case 3 aims to break up these silos. The goal is to develop methods and technologies to collect data from a fleet of vehicles and transmit them to the cloud where they are stored and processed. The collected data can then be used to gain deeper insights by utilizing different cloud services. Further, data can be exchanged with third parties to integrate their services as well.

The figure below shows the mapping between the TRANSACT and UC3 components. Please note, that the Device (the Battery Management Systems) remains unchanged. The system is extended by Edge and Cloud.

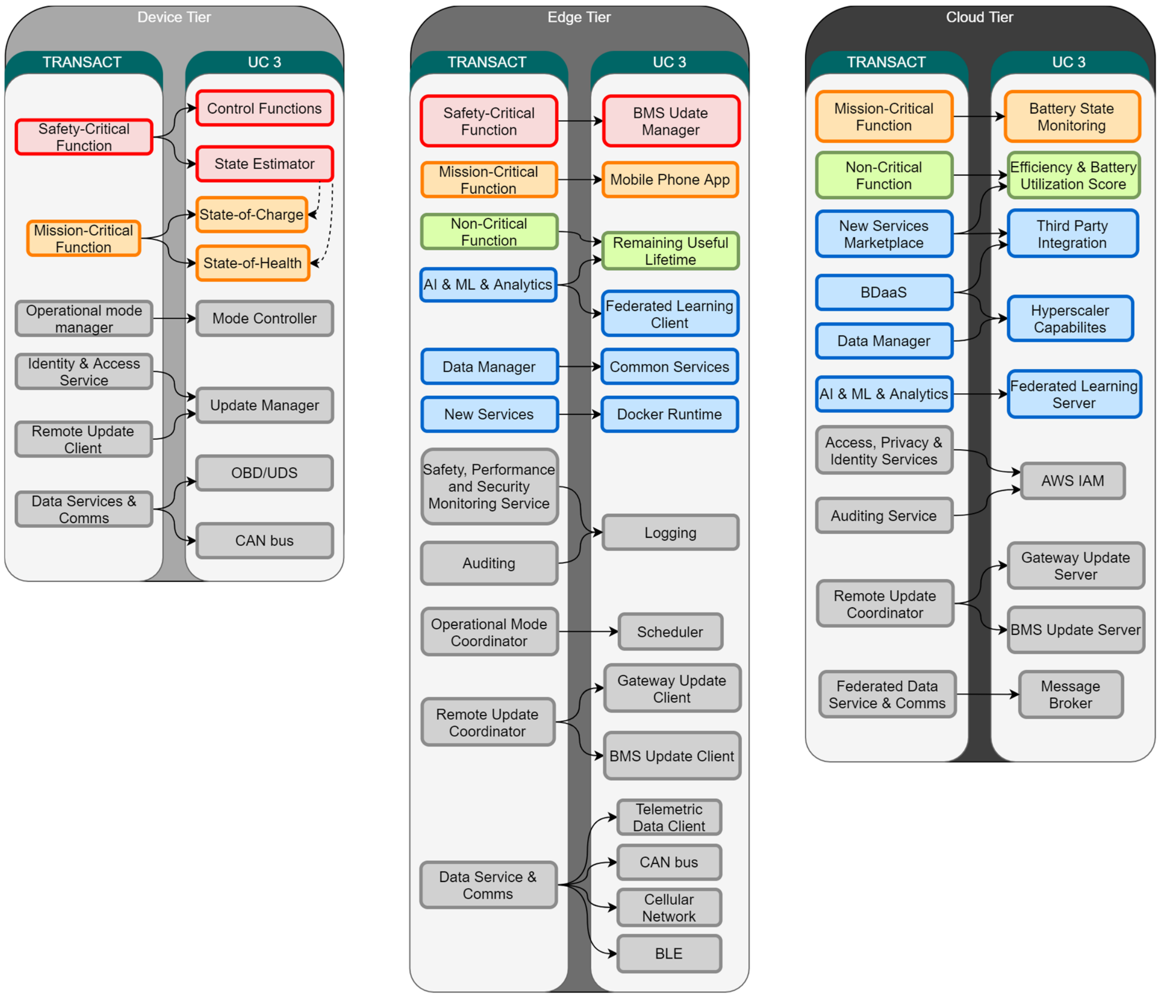


Figure 11: TRANSACT architecture instance in UC3

A major challenge regarding the verification and validation process is the availability of the “device”. A device means in this context a battery electrical test vehicle. Especially when it comes to testing and benchmarking the backend, the tests are limited by the number of vehicles, drivers and the associated personnel.

2.2.4 Use Case 4 - Edge-cloud-based clinical applications platform for Image Guided Therapy and diagnostic imaging systems

The high-level architecture instantiation for use case 4 follows the 3-tier approach as described in the overall TRANSACT architecture. Imaging devices such as X-ray imaging equipment, including safety critical functionalities like real-time X-ray exposure control during live treatment, will remain in the device tier. Mission critical functionalities for diagnosis, planning, viewing and analytics will move towards edge and cloud tiers. Integration of 3rd party applications is anticipated in the UC4 architecture, which will create opportunities to easily integrate their functionalities in the overall Image Guided Therapy solutions

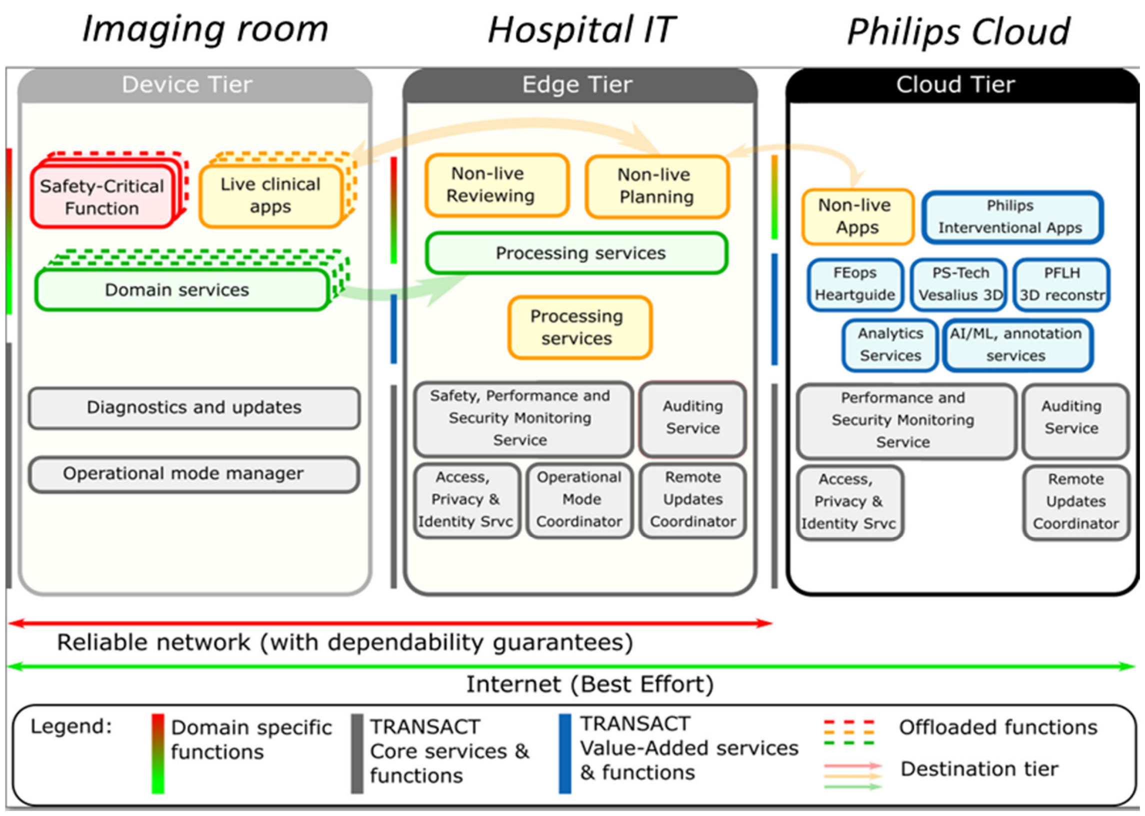

Imaging room
Hospital IT
Philips Cloud
Device Tier
Edge Tier
Cloud Tier
Safety-Critical Function
Live clinical apps
Domain services
Diagnostics and updates
Operational mode manager
Non-live Reviewing
Non-live Planning
Processing services
Processing services
Safety, Performance and Security Monitoring Service
Auditing Service
Access, Privacy & Identity Srvc
Operational Mode Coordinator
Remote Updates Coordinator
Non-live Apps
Philips Interventional Apps
FEops Heartguide
PS-Tech Vesalius 3D
PFLH 3D reconstr
Analytics Services
AI/ML, annotation services
Performance and Security Monitoring Service
Auditing Service
Access, Privacy & Identity Srvc
Remote Updates Coordinator
Reliable network (with dependability guarantees)
Internet (Best Effort)
Legend:
Domain specific functions
TRANSACT Core services & functions
TRANSACT Value-Added services & functions
Offloaded functions
Destination tier

Figure 12: TRANSACT architecture instance in UC4

Quality of Service (QoS) is an important challenge to ensure that mission critical image processing will remain available, since it is being used during clinical interventions, even in case the cloud-based image processing is not performing properly. To address this challenge, the architecture components taking care of performance monitoring/management and operational mode coordination are essential.

2.2.5 Use Case 5 - Critical Wastewater Treatment Decision Supported by Distributed AI

Use Case 5 aims at enriching the functionality of existing devices and Wastewater treatment plants (WWTPs) themselves, leveraging the resources in the edge and in the cloud layers. This implies that the operation of several processes which already exist in the plants will be complemented, aided or assisted by processes that will be deployed in the edge or in the cloud.

An overview of these processes is provided below. The edge layer represents a local deployment in the WWTPs, while the cloud layer represents a common deployment at DAM headquarters. Besides the domain specific functions, there will be some core and value-added services deployed. Namely, at the device layer, the “data services and comms” and the “Safety, performance and security” services will be required in all three WWTPs, while the “Operational mode manager” will depend on the solutions being run in the WWTP.

At the edge layer, we have a similar situation, where the “Data services and comms”, “Safety, performance and security” and “Operational Mode Coordinator” are going to be used. In this layer, the BDAaaS will be deployed, to facilitate, at least, the construction of the dashboards and the exploitation of the AI models. Other tasks, related to data management and storage would be also assisted by this component.

At the cloud layer, though, the core services in use increase, as the “Access, Privacy and Identity” services will complement the “Data services and comms”, “Performance and security”. Similarly, the BDAaaS component will assist in the creation of the dashboards and in the deployment of the software for training the models.

In both the edge and cloud layers, all the TRANSACT architecture services will be run as microservices on a platform that provides the necessary guarantees in terms of their deployment, management, monitoring, updating and maintenance.

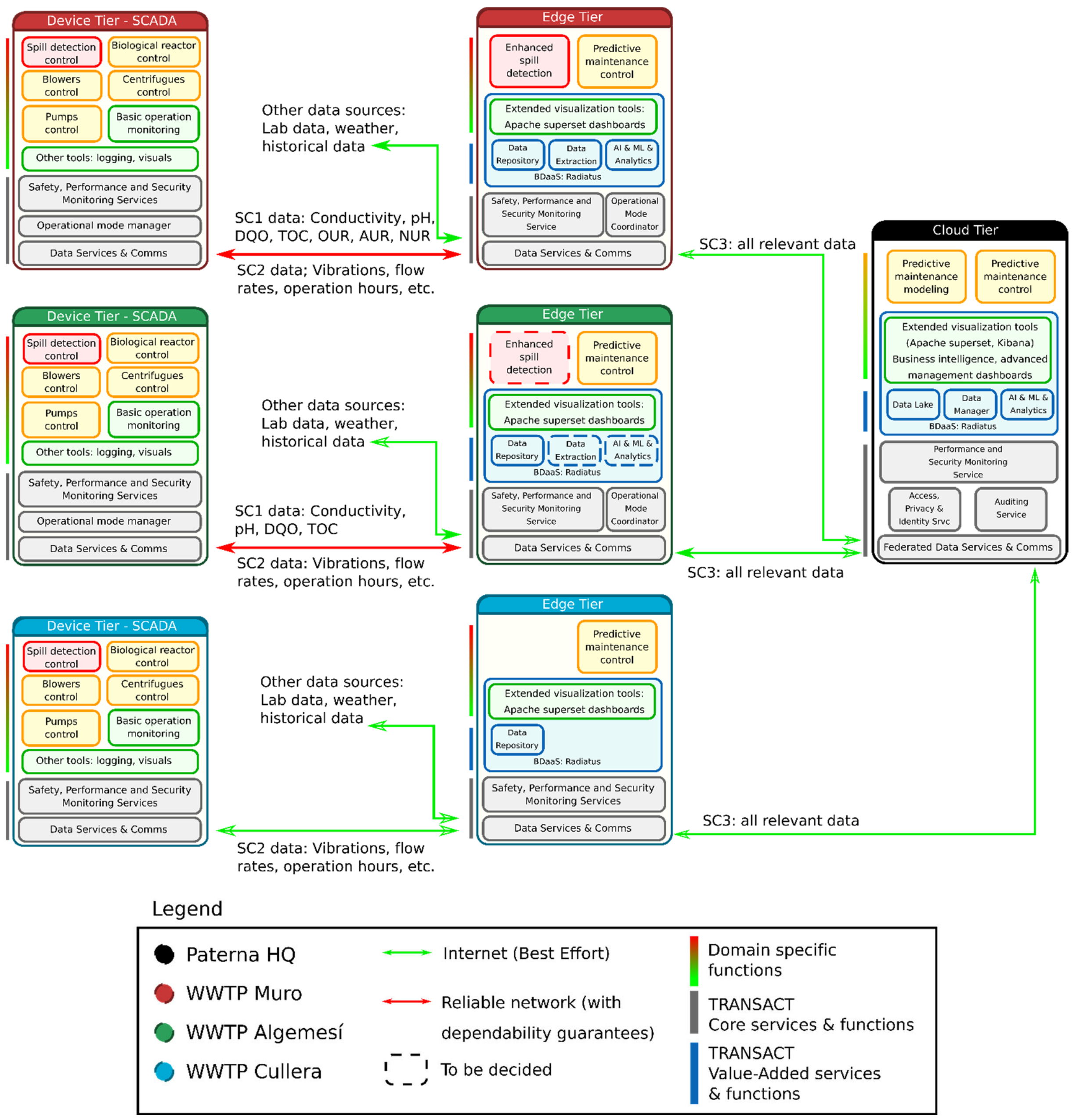

Device Tier - SCADA
Spill detection control
Biological reactor control
Blowers control
Centrifugues control
Pumps control
Basic operation monitoring
Other tools: logging, visuals
Safety, Performance and Security Monitoring Services
Operational mode manager
Data Services & Comms
Other data sources: Lab data, weather, historical data
SC1 data: Conductivity, pH, DQO, TOC, OUR, AUR, NUR
SC2 data; Vibrations, flow rates, operation hours, etc.
SC1 data: Conductivity, pH, DQO, TOC
SC2 data: Vibrations, flow rates, operation hours, etc.
Edge Tier
Enhanced spill detection
Predictive maintenance control
Extended visualization tools: Apache superset dashboards
Data Repository
Data Extraction
AI & ML & Analytics
BDaaS: Radiatus
Safety, Performance and Security Monitoring Service
Operational Mode Coordinator
Data Services & Comms
SC3: all relevant data
Cloud Tier
Predictive maintenance modeling
Predictive maintenance control
Extended visualization tools (Apache superset, Kibana) Business intelligence, advanced management dashboards
Data Lake
Data Manager
AI & ML & Analytics
BDaaS: Radiatus
Performance and Security Monitoring Service
Access, Privacy & Identity Srvc
Auditing Service
Federated Data Services & Comms
Legend
Paterna HQ
WWTP Muro
WWTP Algemesí
WWTP Cullera
Internet (Best Effort)
Reliable network (with dependability guarantees)
To be decided
Domain specific functions
TRANSACT Core services & functions
TRANSACT Value-Added services & functions

Figure 13: TRANSACT architecture instance of UC5

In UC5, model transformations are used to generate different dashboards that collect and represent data from WWTP sensors. These dashboards monitor the WWTP's data and notify the plant operator about abnormal values by raising configurable alarms. These abnormal values can warn about a potential flood, which requires manual operation from plant operators. In this context, it is crucial that sensors' data monitors and dashboards can keep up with the amount of data generated by plant sensors. To ensure this, model transformations can be used to generate different deployment models (which in turn, can generate different deployment recipes) that allow testing whether a specific deployment can guarantee the expected levels of availability and performance.

3 Overview of validation approach

In this section, we outline a generic transition & validation approach aligned with the TRANSACT transition process. This generic transition & validation approach is used as a reference for the applications of the validation and assessment methods of T2.2. The transition & validation approach highlights the verification and validation needs along the transition process and thereby shows where simulation-based assessment should establish certain system (model) properties.

In Section 3.1 first an introduction to the Vee-model is given, then our adaption of the Vee-model for the development of SCDCPS within TRANSACT is presented. We will use this Vee to structure the activities during the system development in the following chapter.

In Section 3.2 we discuss the relation of our Vee and the DevOps cycle as considered in D23(D1.3).

In Section 3.3 we present a generic transition & validation approach. To this end, we describe in a generic way the steps of transforming a system over the edge-cloud continuum along with the resulting need of the verification and validation activities.

3.1 Process for design and validation of SCDCPS

As a basic process for the design and validation of SCDCPS, the TRANSACT partners agreed to use the well-established Vee-model for T2.2. Within T2.2, the Vee-model has been adapted to the special challenges of SCDCPS, as shown in Figure 3. When in Sect. 4 the developed methods & tools are presented, it will be highlighted where in the Vee process the developed methods will be applicable.

We will first briefly introduce the Vee-model in its basic form, and we will then give a detailed description of the extensions to the Vee-Modell, focusing on the implications of transforming a safety-critical CPS into a safety-critical distributed CPS.

As stated in D4 (D5.1) on page 8, "The basic functional safety IEC standard 61508 recommends using the Vee-model approach for software design and development of safety-related systems [1]. The IEC 61508 standard is applicable to most industries and some industries have defined their own safety standards based upon IEC 61508 […]".

The Vee-model describes the software development process as a sequence of development phases within the life cycle of the system and explains the relationship between these phases. The left thigh of the Vee represents the decomposition and refinement of system requirements, where specification, design and development activities finally lead to the specifications of directly implementable components. At the left thigh of the Vee, first the user requirements are specified, then the functional requirements and finally the software design are specified. At the bottom of the Vee, the specified software components are implemented. The right thigh of the Vee represents the step-by-step integration and testing of the implemented components to finally create the verified and validated product. During development, the properties of the product are continuously validated and verified.

Due to the provided guidance for the planning and realization of projects, the Vee-model reduces the project risks, the quality of the product is increased as intermediate steps are checked early on, the total cost over the life cycle of the system is reduced as better tracing and early testing avoid costly late changes, and the communication between all stakeholders is improved.

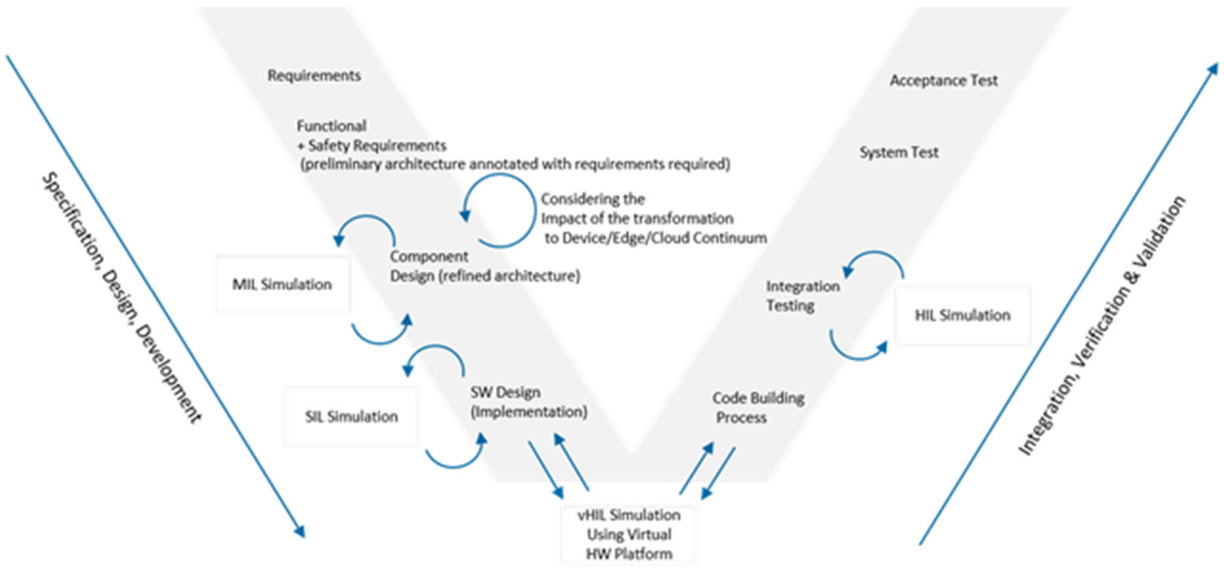


Figure 14: The Vee-model - A Process for Design and Validation

Figure 14 shows the version of the Vee-model that is used by the TRANSACT partners in T2.2. The Vee-model is adapted to the transformation of safety-critical CPS to safety-critical distributed CPS. We next list the extensions to the traditional Vee-model and we describe them in more detail afterwards:

1. Functional as well as safety requirements are derived based on the preliminary architecture of the safety-critical CPS.
2. The impact of the transformation to a distributed safety-critical CPS is then analysed in order to refine the preliminary architecture.
3. The “Model-In-the-Loop” simulation examines the component design. This may trigger further refinements of the architecture.
4. At the bottom of the Vee the virtual hardware “in-the-loop” (vHIL) is used to examine the software design.
5. The code building process is considered as a press-button concept. It is hence located at the right thigh of the Vee.
6. The integration test is done via “Hardware-in-the-loop” (HIL) simulation.

The TRANSACT Vee-model starts with the specification of user requirements, just like the traditional Vee-model. Since TRANSACT assumes that a safety-critical CPS is given, which shell be transformed into a safety-critical distributed CPS, TRANSACT also assumes that the initial system is given along with its functional and safety requirements. These are considered when specifying the functional and safety requirements for the SDCPS under development.

MIL consists of mixing the physical plant model (including mechanical, electrical, thermal, etc., effects) with the control strategy at the algorithmic level (typically using state machines). This creates a complete mechatronic model that can be simulated together to find out whether the behaviour of the control logic is correct. Initial tests can be created at this level. These tests examine whether the component design of the SDCPS meets the requirements.

“Software-in-the-Loop” (SIL) simulation is then used to validate the implementation of the TRANSACT architecture. With SIL, the implementation of software architecture is simulated together with the same physical plant model used in the MIL phase.

The automotive industry developed the concept of HiL simulation, where the physical plant is replaced by a simulation model (a.k.a. the plant model). To connect the ECU and the plant model in a closed loop, the plant model is executed on a HIL simulator box. This HIL simulator box speeds up the simulation of the plant model to real-time. Today's more advanced testing environments use HIL, hence avoiding the risk of damaging the plant, but more importantly allow testing to happen earlier. The dependency on physical ECU prototypes can be removed as well by using fast simulation via a virtual prototype (VP) of the ECU. The goal of vHIL is to frontload the testing process by using a virtual prototype of the ECU hardware before the actual ECU hardware is available. With a vHIL environment, a virtual ECU model is connected in closed loop to the same plant model.

3.2 The V-Model and DevOps

In the TRANSACT transition guide, D23(D1.3), the importance of the cross-cutting concerns (safety, performance, security and privacy, and regulatory and certification) is stressed. Their impact on the product design, development, and operation is outlined and the activities for each concern along the DevOps pipeline (cf. Figure 15) is sketched. Although the TRANSACT system development process is an on-going and agile process, we nevertheless use the Vee-Model to align our activities in T2.2.

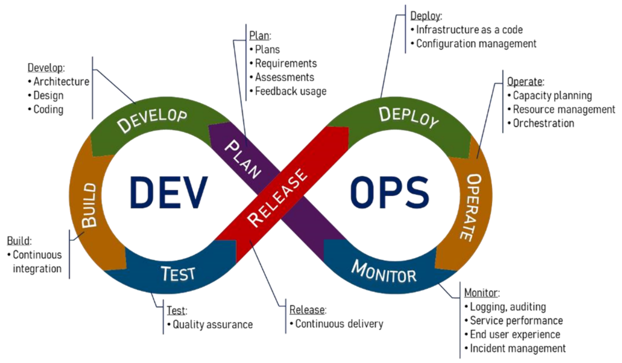


Figure 15: DevOps pipeline as used in D23(D1.3)

The DevOps pipeline emphases the continuous development cycle, where a system evolves as time goes on as a sequence of changes. The DevOps pipeline subsumes all activities described in previous section while the traditional V-Model does neither cover the operation and monitor phase of the DevOps pipeline nor does it cover a planning phase where feedback from the initial/current system is used to define the changed requirements of the next system's version.

The activities of T2.2 concentrate on simulation and assessment of a (SDS)CPS. Figure 16 illustrates how the activities of T2.2 relate to the DevOps pipeline. The focus is "in the traditional sense of the Vee" on examining a system (model/design variant) rather than on the continuous development cycle. With other words, the repeated and ongoing evolution of the system plays only a minor role for us – T2.2 assumes an initial system to be given and focuses on assessing or simulating the newly developed system (variant/model). The Vee-Model hence gives a better fit of the activities that have been treated in T2.2

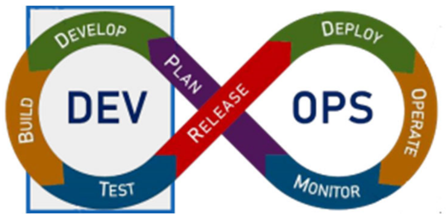


Figure 16: DevOps pipeline and focus of T2.2

3.3 Transition and Validation Methodology (T&V Methodology)

In this section, we outline a general transition and validation methodology, that will be used to highlight where the results of T2.2 are required for the transition process from an initial system to a distributed system.

3.3.1 Motivation

In the following, we describe in a generic way a stepwise process of defining requirements when transforming an initial CPS into a system that gets in parts distributed over the edge-cloud continuum. The methodology starts by defining what is expected from the overall distributed solution and ends with having defined the requirements for all components. The methodology thus defines increasingly more concrete requirements reflecting design decisions that have been taken. When it turns out that a requirement cannot be realized, the design variant will be dismissed and the alternatives will be explored.

The methodology leads to:

1. early identification of functionality that will be offloaded,
2. early check if offloading is feasible at all (e.g., with a view to safety)
3. support of the creation and integration of all interfaces necessary for offloading (including but not limited to creating facades on the device, defining APIs, eliciting requirements on communication media (e.g., with a view to safety, security, privacy)),
4. support of the design of mitigation strategies for the effects of offloading (e.g., fallback to device side functions using mode management or distributing increased latencies due to communication among all the requirements in a systematic way)

By defining a methodology of requirement specification, the T&V methodology also specifies what validation and verification needs arise when a SCCPS *S* gets transformed into a SCDCPS *S'*: Validation and verification approaches must ensure that the developed system *S'* satisfies the specified requirements. To establish as early as possible whether it is plausible that a realization of the requirements can be found, formal abstract system models must be analyzed, and abstract design variants must be simulated. Since an initial concrete system *S* is given, hybrid simulation-based techniques play an important role. The T&V methodology process is not a prerequisite for applying the methods developed within T2.2 but it serves as a means to concretize the resulting needs for simulation-based methods in a systematic and abstract way.

3.3.2 Relation to the TRANSACT transition methodology

The TRANSACT transition methodology (cf. D23(D1.3)) considers the transition process holistically starting from the business perspective. In contrast, in T2.2 the technical system development process is considered to explain where in the development process the methods & tools for validation and assessment of a system (model/variant) are required.

For the technical realization the key cross-cutting aspects safety, performance, security and privacy, and regulatory certification are of particular concern. As outlined in D23(1.2) for each aspect requirements must be specified, dedicated views for the early models must be developed and examined, so that early testing allows for systematic design and early trade-off analysis. These aspects must be reflected by specifying appropriate requirements when applying the T&V methodology. The T&V methodology treats the requirements abstractly and highlights the basic steps of the development of requirements when distributing parts of a system.

3.3.3 Steps of the T&V Methodology

The T&V methodology considers the case that an improved version *S'* of a given SCCPS *S* is to be developed. The focus is on the case where *S'* is derived from *S* by distributing a set of its functions S_d over the edge-cloud continuum. More precisely, we assume that S_d gets replaced by a set of functions S'_d that use the services of the TRANSACT architecture and together with them provide the improved functionality. The T&V methodology leaves open how the exact transition strategy is. For instance, S_d may be rehosted or S_d may get refactored and rehosted.

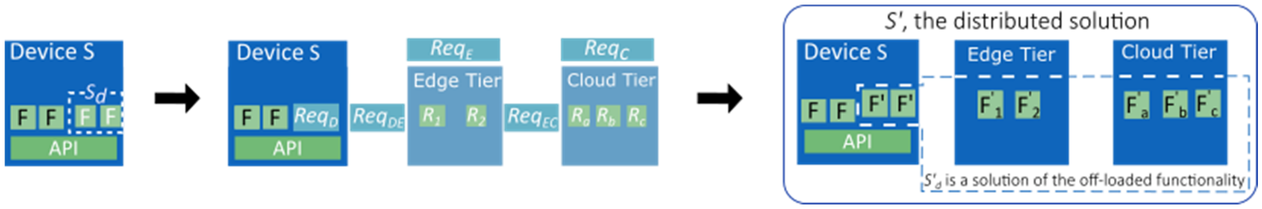


Figure 17: Requirements definition by the T&V methodology leads to a distributed solution *S'*

The basic argument for *"S' is an improved version of S that satisfies the requirements Req'"* is that it has been established that the distributed S'_d *"adequately replaces"* the initial S_d. In order to develop such a distributed version, S'_d, the T&V methodology outlines how requirements can be defined that imply that S'_d will *"adequately replace"* S_d.

The T&V methodology drives the definition of requirements and aims to establish whether the requirements can be realized as early as possible, i.e., by validation and assessment early in the Vee. The tools and methods of T2.2 contribute to this assessment.

Figure 18 gives an overview of the T&V methodology. In the following we explain the methodology in more detail.

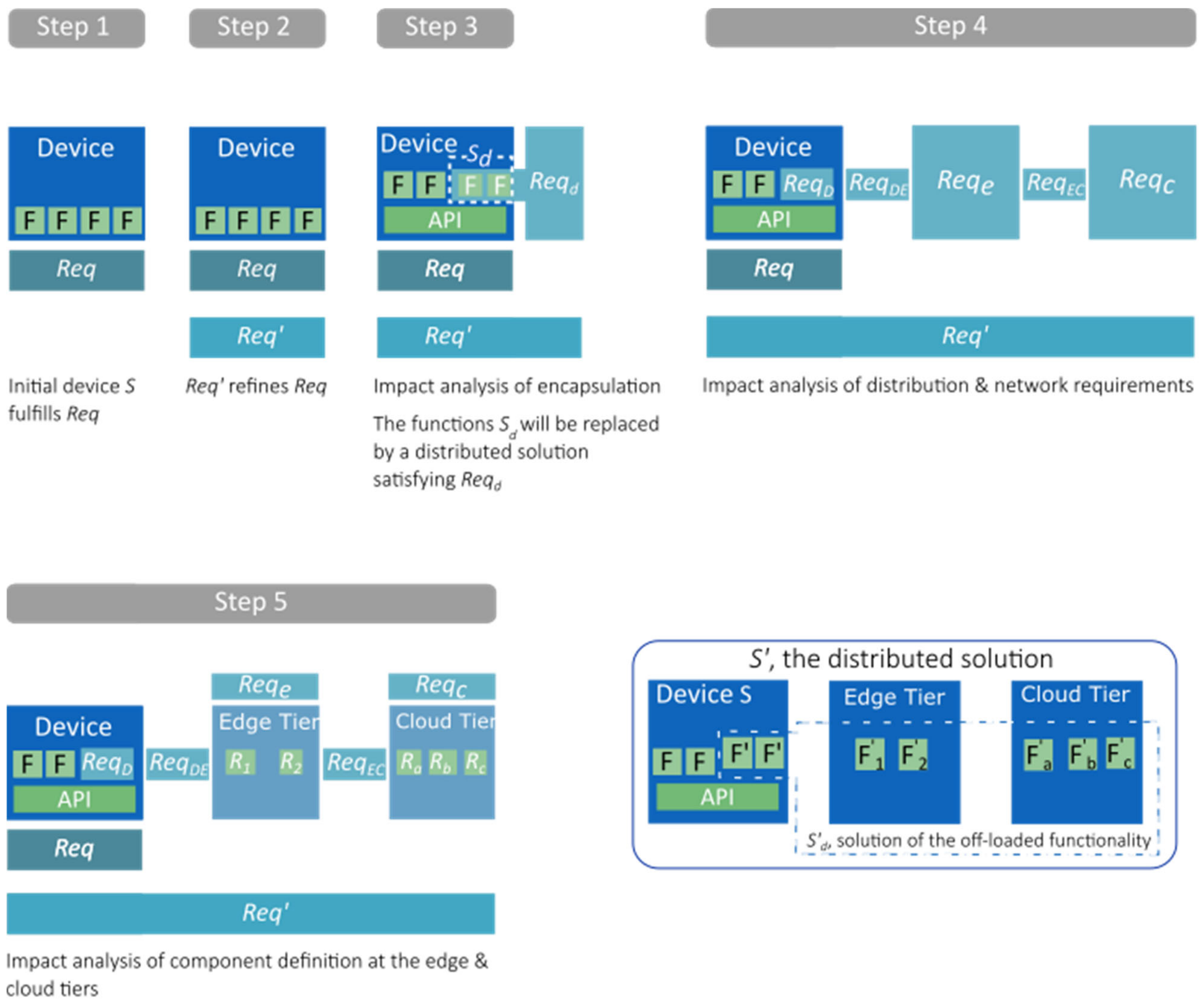

Step 1
Device
F F F F
Req
Initial device S fulfills Req
Step 2
Device
F F F F
Req
Req'
Req' refines Req
Step 3
Device
S_d
F F F F
API
Req_d
Req
Req'
Impact analysis of encapsulation
The functions S_d will be replaced by a distributed solution satisfying Req_d
Step 4
Device
F F Req_D
API
Req_DE
Req_E
Req_EC
Req_C
Req
Req'
Impact analysis of distribution & network requirements
Step 5
Device
F F Req_D
API
Req_DE
Req_E
Edge Tier
R_1 R_2
Req_EC
Req_C
Cloud Tier
R_a R_b R_c
Req
Req'
Impact analysis of component definition at the edge & cloud tiers
S', the distributed solution
Device S
F F F' F'
API
Edge Tier
F'_1 F'_2
Cloud Tier
F'_a F'_b F'_c
S'_d, solution of the off-loaded functionality

Figure 18: T&V Methodology.

The development steps lead to increasingly refined requirements. Critical realizability checks are preferred as early as possible. Simulative methods for early design models therefore play an important role.

In the following, we use decorated versions of *S (S', S'$_d$,…)* and *Req (Req', Req'$_d$,…)*. We usually refer to implemented solutions of requirements (i.e., system or functions) as *S* (plus some decoration) and we refer to the specification of requirements by some decorated form of *Req*. To stress that the implemented solution must satisfy the respective requirement, we try to use matching decorations, i.e., *Req'* is implemented by *S'* (and which is synonymous for "*S'* satisfies *Req'*" in the following*)*.

3.3.4 Stepwise Refinement of Requirements & Exploration of the Design Space

For simplicity, we assume that *S* only consists of an end device and satisfies the requirements *Req*. We also assume that the system *S'* -to be developed- is developed to provide additional or improved functionality.

In **Step 1,** the process starts with a CPS end device *S* on which a collection of functions is implemented. The device is known to satisfy the requirements *Req* since they were established during its design.

In **Step 2**, the new requirements *Req′* are specified. As we are interested in service improvements, the requirements *Req′* refine the requirements *Req* of the initial device *S*.

In **Step 3**, the question "How can *S* be modified in order to satisfy the new requirements *Req′*?" is considered. The different design alternatives will be evaluated in terms of cost-benefit considerations, as well as the risks of changing established components. If the new requirements *Req′* cannot be fulfilled locally, a plausibility check will evaluate whether it is feasible to distribute certain services over the edge-cloud-continuum. Figure 19 **Error! Reference source not found.** gives an overview of Step 3 for the case that S_d gets distributed. In the following, we explain step 3 for offloading S_d in more detail.

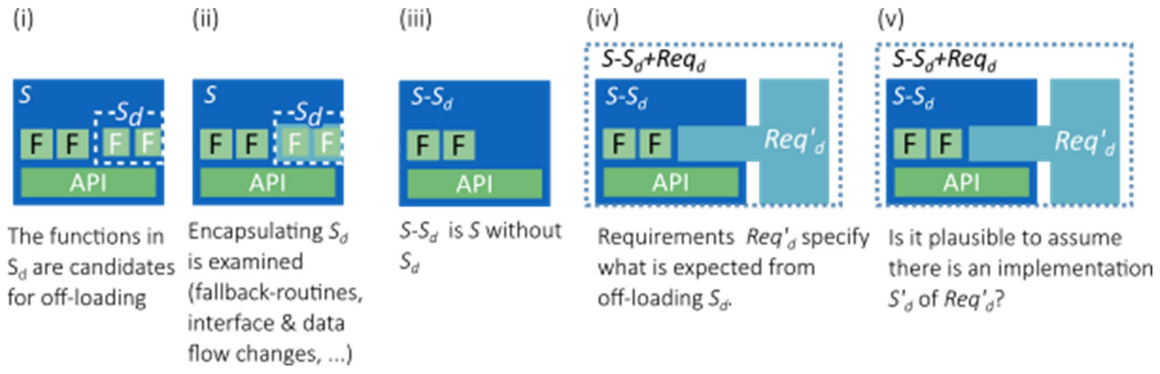

(i)
(ii)
(iii)
(iv)
(v)
S
-Sd-
F F F F
API
S-Sd
S-Sd+Reqd
Req'd
The functions in Sd are candidates for off-loading
Encapsulating Sd is examined (fallback-routines, interface & data flow changes, ...)
S-Sd is S without Sd
Requirements Req'd specify what is expected from off-loading Sd.
Is it plausible to assume there is an implementation S'd of Req'd?

Figure 19: Off-loading the set of functions S_d

Figure 19 illustrates that a set of functions S_d is identified that are good candidates for offloading. S_d is used as a starting point to specify the requirements on the offloaded functionality, Req'_d. Req'_d (cf. Figure 19(iv)) specifies what the future off-device solution S'_d (cf. Figure 17, right most) must provide for the end-device, so that *S′*, the final system, satisfies the strengthened requirements *Req′*.

A plausibility check will answer whether a distributed solution for Req'_d might be found. The cost-benefit considerations will have a dedicated focus on the safety, security and privacy risks associated with offloading S_d.

Since offloading entails additional communication and synchronization for the end device to connect to the edge-cloud continuum, (a) the services S_d will have to be detached from their old infrastructure and (b) a preliminary interface between the end device and the to-be-developed solution S'_d is specified.

Step 3 does not consider concrete solutions S'_d. Its goal is to establish that the new requirements *Req′* are fulfilled when *S* without S_d (i.e., $S\text{-}S_d$) will be equipped with some distributed solution S'_d fulfilling Req'_d. With other words, step 3 does not answer, how Req'_d will be implemented but it evaluates the influence of connecting an abstract encapsulation providing the functionality Req'_d to the end device and checks whether it is plausible to assume that later in the development process an implementation S'_d can be found that realizes Req'_d. Therefore, the approach examines the composition of

1. $S\text{-}S_d$, the end-device after S_d has been stripped off and which is equipped with the TRANSACT components of the end-device, and

2. Req'_d which represents abstractly all possible distributed solutions S'_d.

As one activity of encapsulating (cf. Figure 19(ii)), the impact of offloading on safety, privacy and security is evaluated. Modes and fallback routines on the end-device are specified for the functionality of Req'_d that refers to safety or mission-critical functionality.

If the realization of Req'_d is not plausible, the process steps back to determine a different set S_d of services to be offloaded.

The goal of **Step 4** is an impact analysis of the networks on the distributed solutions for Req'_d. Any distributed solution of Req'_d must cope with the different service levels of the two network types of the TRANSACT architecture: the reliable network between device and edge tier, and the less reliable internet connection that is available between all tiers. In step 4, therefore the possible candidate solutions are examined for their requirements regarding the communication services. The requirements of the candidate solutions are specified as abstract as possible to separate concerns. To this end, the requirements on

- the network connection between device and edge tier Req_{DE},
- the services that will be at the edge tier Req_E,
- the network connection between edge tier and the cloud tier Req_{EC} and
- the services that will be at the cloud tier Req_C.

are defined.

Still no concrete implementation of the services on the edge tier (i.e., a solution of Req_E) or an implementation of the services on the cloud tier (solution of Req_C) is required. However, by defining Req_E and Req_C it is specified what functionality will be realized where. This concretizes the data flow to some degree: What data is necessary at the cloud to realize Req_C and what data is necessary at the edge to realize Req_E. Hence, in Step 4 also security and private aspects will be (re)evaluated.

For assessing whether it is plausible that an implementation of Req_{DE}, Req_E, Req_{EC}, Req_C can be found, the available network providers and the types of networks are considered. It will be taken into account that Req_C will have to be satisfied by the Internet while Req_{DE} can be realized by a combination of the Internet and a more reliable network. One focus in this step is on timing requirements and reliability. As a result of a thorough assessment, the initial versions of the fallback routines designed at Step 3 may have to be adjusted and the according changes in the design must be propagated.

In **Step 5**, the edge requirement Req_E and the cloud requirement Req_C are further refined into component requirements. In Figure 18 we illustrated an example where Req_E is refined by the composition of requirements Req_1 and Req_2, while Req_C gets refined by the composition of the requirements Req_a, Req_b and Req_c. These refinements concretize how the functionality will be implemented on the edge and cloud using the services available at the respective tier.

If no plausible refinement into component requirements can be found, the process steps back to step 4 (or earlier steps). Stepping back to Step 4 allows a new assignment of functionality to the off-device tiers. Stepping back to Step 3 allows to off-load a different set of functions, stepping back to Step 2 allows to modify the targeted functionality Req' and stepping back to Step 1 would allow to exchange even the initial system.

This process of refining Req_E and Req_C to component requirements is often guided by the initial implementation of the functions F_{iii}, F_{iv} on the end device and by the added-value services on the respective tiers, that together can be used to realize the requirements. As in Step 4, the required service level has to be ensured. Therefore, mode management exploiting fallback mechanisms will be used. This might lead to additional communications, so that stepping back to one of Step 4,3,2 or 1 may be necessary.

The Steps 1 to 5 together build a strategy to explore the design space for creating a distributed solution. After Step 5 has been successfully completed, the requirements for all components have been specified. In the best case, the specified requirements allow to derive their implementations via a press button approach.

3.3.5 From Requirements to a Solution -- Verification and Validation

The above process describes how a monolithic pure on-device solution is (partly) distributed to cloud and edge and how requirements for the distributed solution can be derived based on the original solution's requirement and how the requirements can be (re-)established.

The process is not intended to specify a strict temporal order in which the respective requirements are further concretised. The order can be based on the need to analyse critical design decisions. Rather, the T&V methodology outlines the key design steps that lead to a distributed solution and guides the exploration of the design space.

Since it must be shown that the final system fulfils the requirements and the complexity of the system is immense, the T&V methodology can benefit from the use of contract theories. We can think of the requirements as contracts. Consider a contract theory that defines composition, quotient and refinement of contracts. If we have shown that a contract *C* is refined by the composition of contracts C_1 and C_2, then the further evolution (refinement to final implementation) of each C_i can be done independently of the others. Contract theory guarantees that contract *C* is fulfilled. When using contracts, it is sufficient to prove each refinement relation. Thus, it can be avoided to analyse the final implementation of *S'* as a monolithic system, instead the abstract contracts $C, C_1, \ldots$ are examined, and the C_i only refer to parts of the overall system. This allows a large reduction of the complexity. Although there is a wide range of formal contract theories, not all aspects of a system are covered. Properties such as reactivity or security, for example, are such properties. This means that there is no mathematical theory that defines how a monolithic requirement *Req* can be decomposed into more concrete requirements $Req_1, \ldots, Req_n$ so that the composition of solutions for $Req_1, \ldots, Req_n$ is guaranteed to be a solution for *Req*. Consequently, (i) structured, expert-based methods such as Risk Storming must be applied to define the requirements during the design process, and (ii) holistic simulation approaches are important to verify that these requirements are met. It is not enough to test a single component, but the interaction of the solutions must be thoroughly tested, also for emergent behaviour.

4 Methods and tools for simulation-based design and assessment

In this section the developed methods & tools of T2.2 are presented in detail.

4.1 V&V of Backend Architecture (UC3)

4.1.1 Overview

The transition of the local CPS to the distributed CPS introduces two additional tiers: Edge and Cloud. One can assume that for the existing local CPS test frameworks already exist. However, new test tools are required for the emerging tiers. The Cloud system provides the backend for communication and data handling. Core features of the Cloud Backend are horizontal and vertical scalability, enabling that many devices can connect to the same backend. This includes:

- inbound and outbound interfaces,
- data handling,
- data analytics,
- data storage, *and*
- data visualization.

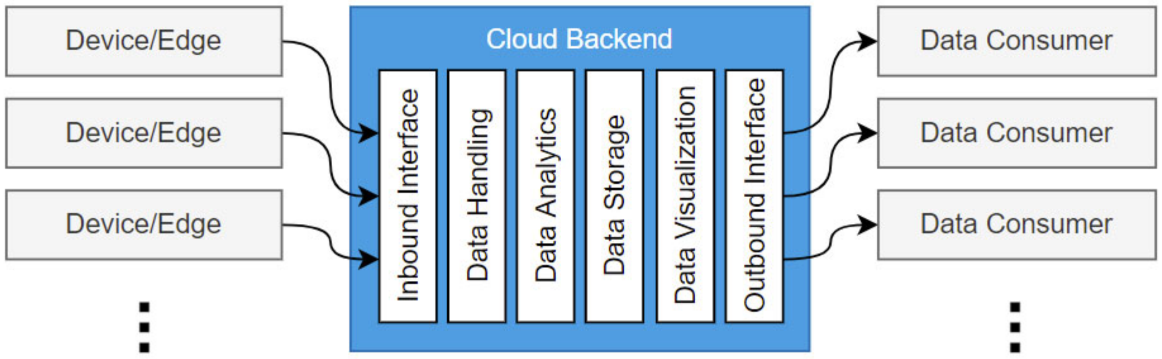


Figure 20: V&V of Backend Architecture.

While simulations of the architecture can provide insights into the system behavior when scaled, they may be based on parameters which are not known and also quite time consuming to set up. Accordingly, simulations can be too theoretical or have too many degrees of freedom and do not provide enough reliable results for validation.

Another approach is to virtualize the device and edge, such that it can be scaled vertically at only little costs. This enables to put the backend into the loop while lifting the dependencies from device, edge but also third-party integrations (consumers). This approach is cost efficient, reliable, available at any time, scalable and provides the most realistic environment and operational conditions compared to the intended real-world utilization of the backend.

In UC3 (Cloud-Featured Battery Management System), the developed tool simulates the whole vehicle and simultaneously utilizes the Telemetric Client Software stack. The Cloud Backend cannot distinguish between real and virtual vehicles and hence the tool provides a very realistic operational environment. Therefore, a framework to simulate a whole vehicle, which produces the same data as a real vehicle, has been developed. The output is a stream of raw CAN frames as the control units of the vehicle would send. The CAN frames are then fed into the Telemetric Client software, which is also virtualized and can run either on the target hardware or in a Docker container in a runtime environment. This enables scalability while no individual hardware per client is required.

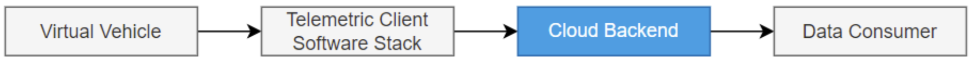


Figure 21: Validation and Verification of the cloud backend architecture in the loop.

The Cloud Backend not only consumes data, but also provides data to other (third party) services which can run in a separate cloud. The functionality of the Cloud Backend as data distributor needs to be verified and validated as well.

The developed Virtual Vehicle framework is a lean approach dedicated for the purpose of V&V. In contrast to the Cruise.M tool, which AVL also owns, time savings are tremendous as setting up a vehicle is quite time-consuming. Cruise.M is used to simulate the physical behavior of vehicles on a very detailed level. The high degree of physical reality is paid with a high demand of computational and personnel resources.

4.1.2 Application of the V-Model

The development process in UC3 follows the V-Model. However, development and integration are not following a strict and single top-down design with a bottom-up or “big-bang” integration process. An agile feature driven development process is preferred. This means that the V-Model is applied iteratively. Due to the fact that many partners contribute to UC3 which deliver contributions at different pace, components are integrated when available. Hence, two stages of V&V are applied: Continuous V&V during the development phase and a final V&V before releasing the final demonstrator. The Virtual Vehicle framework can support both phases. In the following, examples are given for the first case, as the final V&V process takes place after the delivery of this document.

4.1.3 Application in UC3

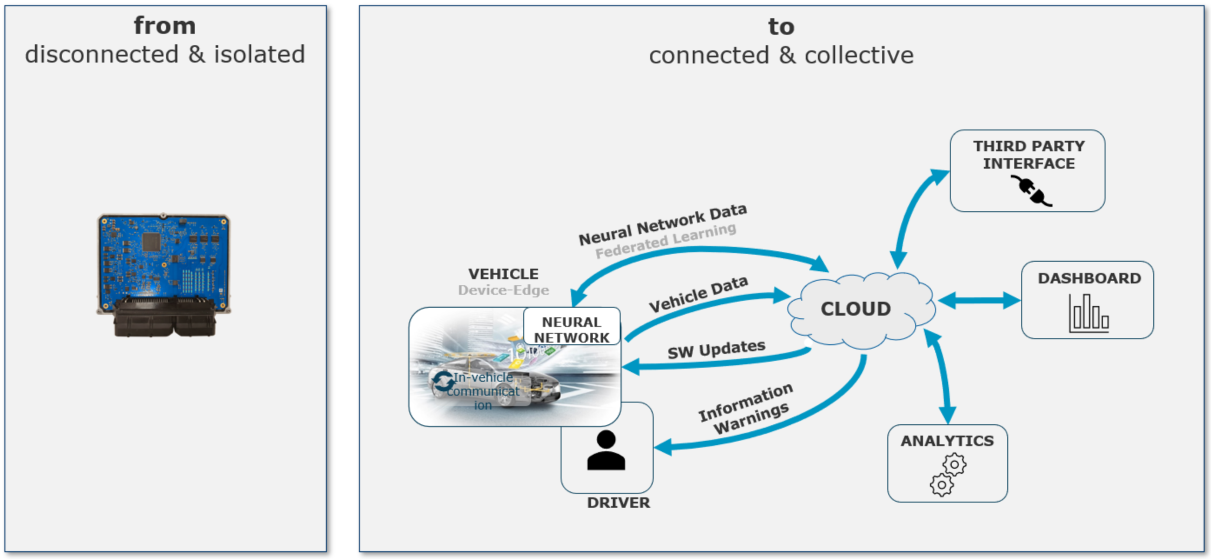


Figure 22: From disconnected to connected & collective systems in UC3

UC3 extends the isolated Battery Management System of a Battery Electrical Vehicle to a distributed system. This enables the capability to monitor the fleet of vehicles by transmitting vehicle and battery data to the cloud backend. The visualization of the data but also the analysis of time stream and session data enables timely interventions and provision of safety-relevant information.

The Virtual Vehicle is used to substitute real vehicles to support fast development of the downstream components, mainly the cloud components. In this way, no real vehicles are required for developing, designing, and benchmarking the cloud backend.

4.1.3.1 Data Pipeline – Workload and Timing

The Figure 23 shows the computation of the individual cell resistances of the 180 cells of the test vehicle. The analysis was performed only for the last third of the available data points, due to a flaw of the backend architecture. The developers figured out that a wrong setting of an AWS service led to this behavior. After changing the parameter, the backend required another V&V loop.

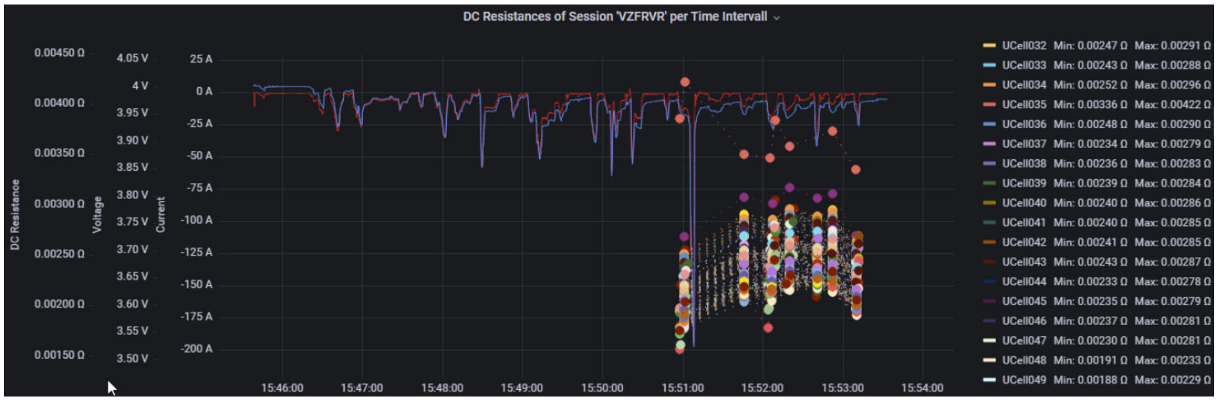


Figure 23: Missing computed data points due to a flaw in the architecture.

To test the architecture again the Virtual Vehicle was used. The Figure 24 shows data created by the Virtual Vehicle with the fixed issue.

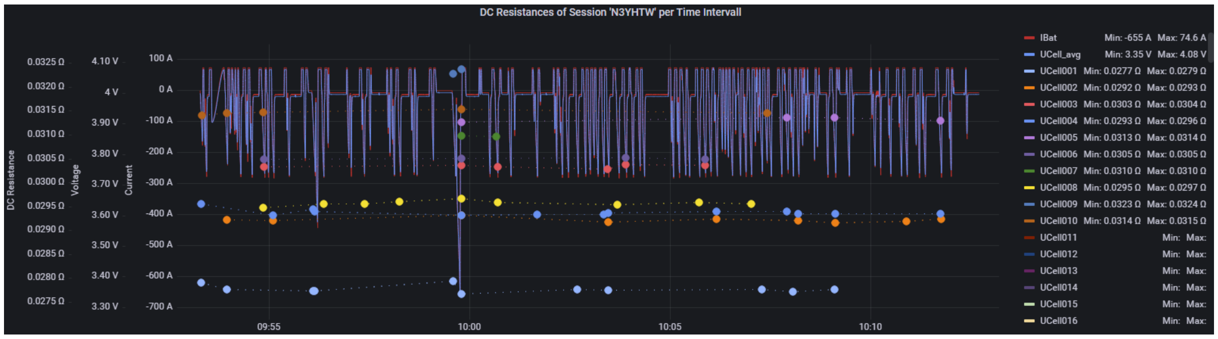


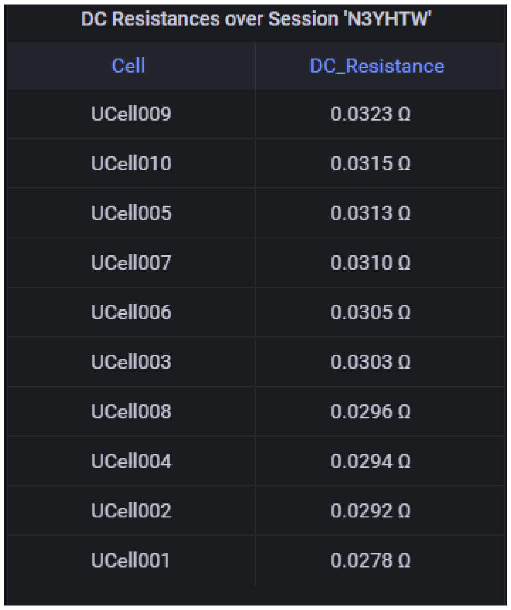
Figure 24: Verification of backend architecture after the fix.

Note that without the Virtual Vehicle, it would be necessary to perform another test drive after identifying and fixing the issues to test the architecture again. Accordingly, a development and V&V cycle would be extremely time consuming.

4.1.3.2 Validation of Data Analytics Results

Figure 23 and Figure 24 show results of an algorithm which computes the individual cell resistances (data points). To determine the values with a higher precision, a statistical approach is applied on top the computed values. However, the cell resistance is in reality not directly measurable and accordingly the validation of this algorithm is almost impossible when a real battery is used. For precise measurements and determination of the cell resistance, electrical impedance spectroscopy (EIS) is the state-of-the-art approach. However, this kind of measurement can only be performed in lab conditions and is quite expensive (in the range of 10-100k€). Accordingly, benchmarking an algorithm is quite challenging.

The situation changes with the utilization of the Virtual Vehicle, because the resistance per cell is a parameter which can be defined. Based on this, the same physical entities which are available in the real scenario are derived. By adding some additional noise, the artificial data are very similar to the real data. When the algorithm is applied, it is now possible to validate the result as the ground-truth value for the resistance is exactly known. Figure 25 shows the results of the computed cell resistances.


DC Resistances over Session 'N3YHTW'

Cell	DC_Resistance
UCell009	0.0323 Ω
UCell010	0.0315 Ω
UCell005	0.0313 Ω
UCell007	0.0310 Ω
UCell006	0.0305 Ω
UCell003	0.0303 Ω
UCell008	0.0296 Ω
UCell004	0.0294 Ω
UCell002	0.0292 Ω
UCell001	0.0278 Ω

Figure 25: Results for the cell resistances after applying a statistical procedure.

The generation of artificial but well-defined data helped to determine an accurate parameter set for the algorithm but also to identify logical mistakes during early development phase.

4.1.4 Summary

The concept of the Virtual Vehicle for verifying and validating the backend architecture was very helpful by frontloading the tests to an early stage of the design and implementation phase. In general, this tool can be used for any V&V process which requires continuous availability of data with well-defined characteristics.

4.2 Risk Storming

The following section describes risk storming, a method that seems very valuable for the transition process and worthwhile to apply in practice.

4.2.1 Overview

There are various interpretations of the term risk, depending on the scope. Safety: The risk that someone gets harmed. Security: The risk that a system becomes a victim of a cyber-attack. Whereas the latter one also has an impact on safety. There is no safety without security.

In the automotive domain, ISO 26262 („Road vehicles – Functional safety“) provides information and guidelines on how to treat functional safety of electrical and/or electronic systems that are installed in road vehicles. Analysis, assessment and the definition of safety goals is subject matter of dedicated safety experts. Noteworthy is that there are well defined and established architectural patterns (e.g., AUTOSAR).

TRANSACT aims to extend the system to the cloud, accordingly the standard is not applicable anymore. SW architecture in the cloud/edge does not follow a given design pattern anymore. This could lead to the bad idea that you can change the design also in a later stage of the development, which sometimes is mysteriously called agile. Frequent changes, adaptations and extensions of the architecture can lead to the anti-pattern which is called the “big ball of mud” - a chaotic error-prone software architecture. Non-functional requirements are mainly the driver for the software architecture design and need at least as much attention as the functional requirements.

The paradigm on how to develop software in the cloud shifts dramatically; the scenarios which could lead to failures are completely different. Nevertheless, there is a need to identify and assess risks. The MAGERIT[2] approach comes with a pre-defined catalogue of possible scenarios, with the possibility to extend the catalogue manually.

Risk Storming is an approach that solely relies on the experience of software architects, developers and end-users and aims at creating a solid and resilient architecture.

4.2.2 Application of the V-Model

Risk Storming is performed during the design phase at the architecture level. The steps are as follows:

1) Draw the preliminary software architecture diagrams. You can use, e.g., UML or the C4 notations style.
2) Identify the risks individually: Every participant individually analyses the architecture and writes sticky notes which contains the following three information: A description of the identified risk, and two numbers to assess the probability and the impact, each between 1 and 3. The color of the sticky note represents the overall criticality.

   Examples for risks can be: Data formats from third-party systems change unexpectedly, external systems become unavailable, components run too slowly, components don't scale, key components crash, single points of failure, data becomes corrupted, infrastructure fails, disks fill up, ….

   The overall criticality is computed by multiplying probability and impact, leading to the color code for the sticky notes:

Impact\Probability	Low - 1	Medium - 2	High - 3
Low - 1	🟩 1	🟩 2	🟧 3
Medium - 2	🟩 2	🟧 4	🟥 6
High - 3	🟧 3	🟥 6	🟥 9

Figure 26: Criticality Colour Codes

3) Converge the risks on the diagrams: Every participant adds the sticky notes to the corresponding components where the risk was identified.
4) Review and summarize the risk: Review and summarize the output, especially focusing on risks that only one person identified, or risks where multiple people disagree on the priority.

[2] Methodology of Analysis and Management of Risks of Information Systems, https://www.enisa.europa.eu/topics/risk-management/current-risk/risk-management-inventory/rm-ra-methods/m_magerit.html

4.2.3 Summary

Risk Storming is a creative, experience-based and collaborative method to identify risks in the context of software architecture. In this way, painful and critical changes of the architecture at a late stage, or even during utilization phase, can be avoided. The result can give a more resilient design which fits best to the non-functional requirements.

4.3 Simulation-based V&V concept for Maritime Advisory Service Framework (UC2)

4.3.1 Overview

The figure below illustrates the implemented framework for performing V&V in UC2.

The generation of the two reference routes in the Figure 28 represents the first test-cycle. The basis of the scenarios for performing these activities are the 383 routes between the globally distributed ports of the statistical analysis. These routes form the basic ground truth, which is underpinned by the cleaned and processed AIS data. The resulting virtual routes, together with the virtual ports and vessels, are stored as scenarios in the scenario database.

Test-cycle 2 is currently still in progress and serves to validate the NAVTOR Auto-Route against the HAGGIS world model.

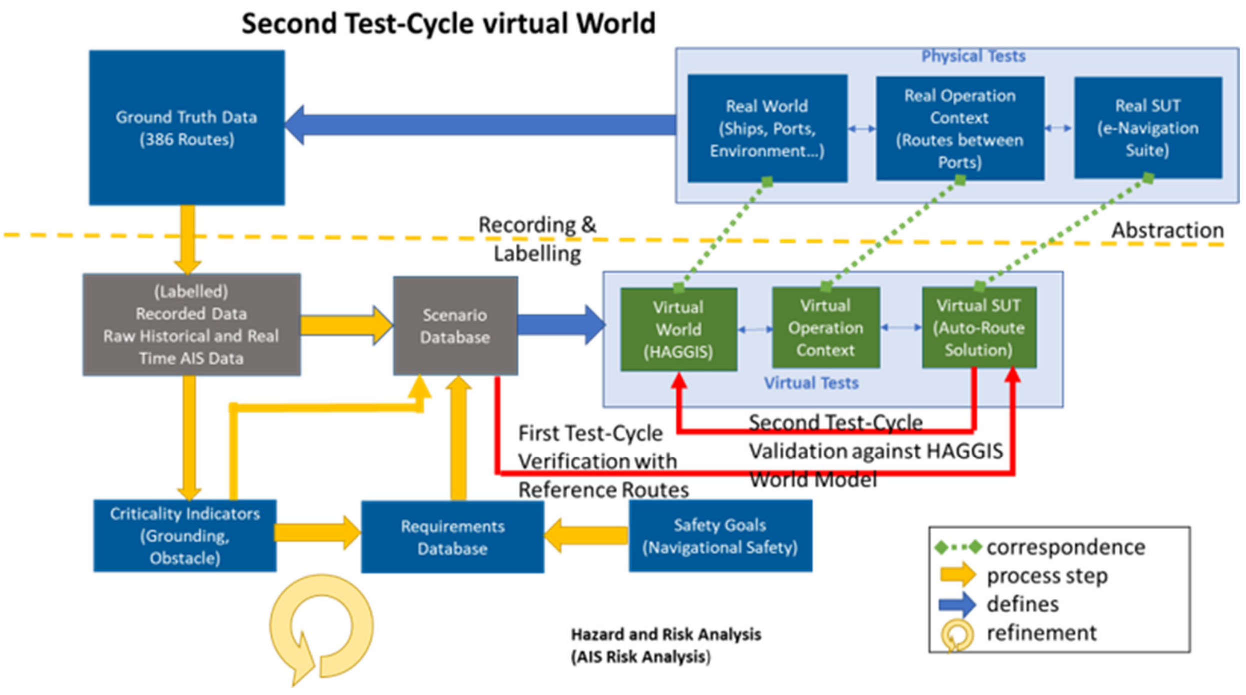

Second Test-Cycle virtual World
Physical Tests
Ground Truth Data (386 Routes)
Real World (Ships, Ports, Environment...)
Real Operation Context (Routes between Ports)
Real SUT (e-Navigation Suite)
Recording & Labelling
Abstraction
(Labelled) Recorded Data Raw Historical and Real Time AIS Data
Scenario Database
Virtual World (HAGGIS)
Virtual Operation Context
Virtual SUT (Auto-Route Solution)
Virtual Tests
First Test-Cycle Verification with Reference Routes
Second Test-Cycle Validation against HAGGIS World Model
Criticality Indicators (Grounding, Obstacle)
Requirements Database
Safety Goals (Navigational Safety)
Hazard and Risk Analysis (AIS Risk Analysis)
correspondence
process step
defines
refinement

Figure 27: V&V Concept of Auto-Routes for seagoing vessels

The following criteria have been defined as safety goals to assess the basic navigational safety of the generated NAVTOR Auto-Routes for vessels:

1. The route must be accessible via the advisory service and follow the route between the two specified ports.
2. The blockers used to establish navigational safety must work.
3. The autoroute must include the respective prescribed TSS and these must be planned in the correct direction.
4. The route must include the relevant prescribed deep- water routes, if required by the draught of the vessel, and these must be planned in the correct direction.
5. The route of the vessel must follow the fairways in the correct direction. The buoyancy must be followed in accordance with the regulations.
6. The generated NAVTOR Auto-Route should not deviate more than 10% in distance from the most common used reference route for a given vessel type (fuel saving and reduction of CO2 emissions).
7. The advisory service should be stable and reliable.
8. Failure of the advisory service should be clearly indicated to the user.

Particularly critical indicators are the avoidance of ground contact and contact with objects such as buoys or moorings.

Our framework allows two test cycles to be performed: In the first test cycle, the NAVTOR Auto-Route is manually compared with the Reference Route of a given vessel and the most frequently used route between the two respective ports. This comparison requires the availability of up-to-date paper charts or ENCs. The main purpose of this test cycle is to verify the requirements.

An essential key to the generation of reference routes is the preparation of AIS data. As this is mainly done manually combining various methods, it is therefore very time consuming. The most frequented Reference-Route should be fundamentally shorter on average than the NAVTOR Auto-Route, as this route is also used by ships with a lower draught. These ships sail the route between the two defined ports closer to the coasts and are therefore shorter. The Reference-Route of the specified vessel will be longer on average than the calculated NAVTOR Auto-Route, as we have generated this route from the AIS data of the real vessel with its current draught. These ships sail deep-water routes that are further away from the coasts, thus increasing the distance between the two ports.

During the second test cycle2, which is currently still under development, the HAGGIS maritime simulation of the German Aerospace Centre is used to validate the generated NAVTOR Auto-Routes. The HAGGIS virtual co-simulation infrastructure includes AI-based environmental, traffic and vessel simulators. These will be used to assess the risks, efficiency and navigational safety of the NAVTOR Auto-Routes for ships during product development.

4.3.2 Challenges of V&V activities

A particular challenge within our V&V activity was to achieve a high level of trustworthiness for the two Reference Routes.

To achieve this goal, the reference route must be of very high quality and reliability. We achieved these properties by using different mathematical models to create the reference route, which are independent of each other and partially overlap.

Moreover, different models were used for the same task and their results were compared and evaluated using the latest techniques. The first step is to extract the relevant data for the route between 2 ports from the available historical AIS data. This step leads from a decision boundary to a validity region. This is our ROI (Region of Interest) (cf. figure below).

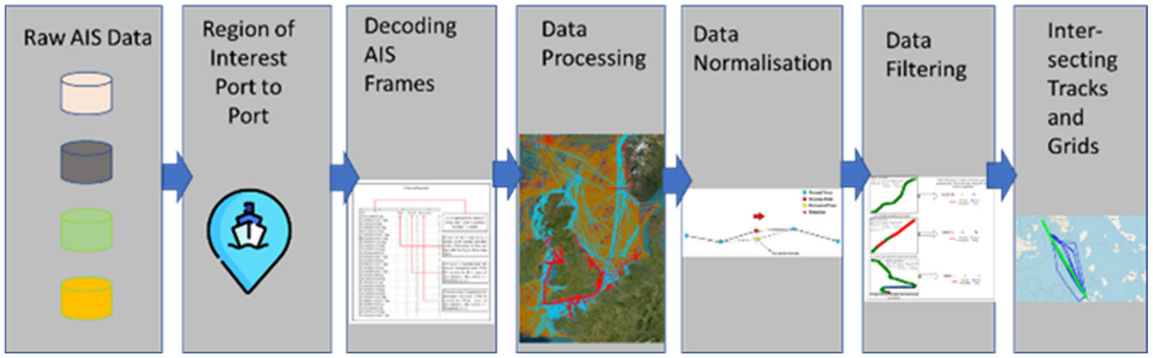


Figure 28: General Process for Deployment Reference Routes

Pre-processing is a basic step performed at the beginning to improve the quality of the trajectory data and to generate sub-trajectories. This process consists of the following three steps:

1. Trajectory extraction and separation: This step is divided into two sub-steps: separation of data from different vessels and separation of data from different trajectories of the same vessel. Vessels are identified by their maritime mobile service identity (MMSI). The trajectory of the vessel is discontinuous in different time periods and different trajectories of the same vessel can be separated according to the time stamp of the AIS data.
2. Time interval standardization: AIS data transmission time interval has a standard specification; however, a large number of AIS data acquisition time intervals do not conform to the standard.
3. Data cleaning: Longitudinal and latitudinal data of ship AIS are derived from GPS (Global Positioning System). The relative positioning of GPS is subject to errors due to atmospheric delay, multipath and diffraction. Data cleaning aims to discard impossible positions or trajectories using specific constraints.

4.3.3 Results

With the implemented V&V architecture, 93 of the 386 statistically determined routes have been checked so far. The summarized results are as follows:

1. The calculation of the NAVTOR Auto-Route is stable.
2. The blockers used to establish navigational safety work perfectly.
3. Most of the NAVTOR Auto-Routes are shorter than the Reference-Route of the specified ship. This reduces CO2 emissions and lowers fuel consumption.
4. Some of the calculated NAVTOR Auto-Routes are longer than the most frequently sailed Reference-Route. However, the distance differences at long distances (10,000 nm) are very small.
5. The NAVTOR Auto-Routes are fundamentally safe from a navigational point of view. The route follows the fairways and passes the buoys on the correct side. Danger spots are effectively avoided.
6. A comparison of the time taken to plan a conventional route (including a safety check) and using a NAVTOR Auto-Route with a safety check showed a saving in planning time of up to 70% (manually stopped times within corresponding comparisons).

The variance of the worldwide route guidance achieved by the static analysis is extended by the retrieval of so-called special routes. These special routes contain special challenges in terms of further comparison possibilities and navigational peculiarities.

4.3.4 Application within System Development

The simulation-based V&V concept for the maritime advisory service framework offers the opportunity to evaluate the system via model in the loop simulations as well as software in the loop via the virtual testbed hybrid architecture HAGGIS and, thus, realizes the front-loading of the Auto-Routes. This is illustrated in Figure 29.

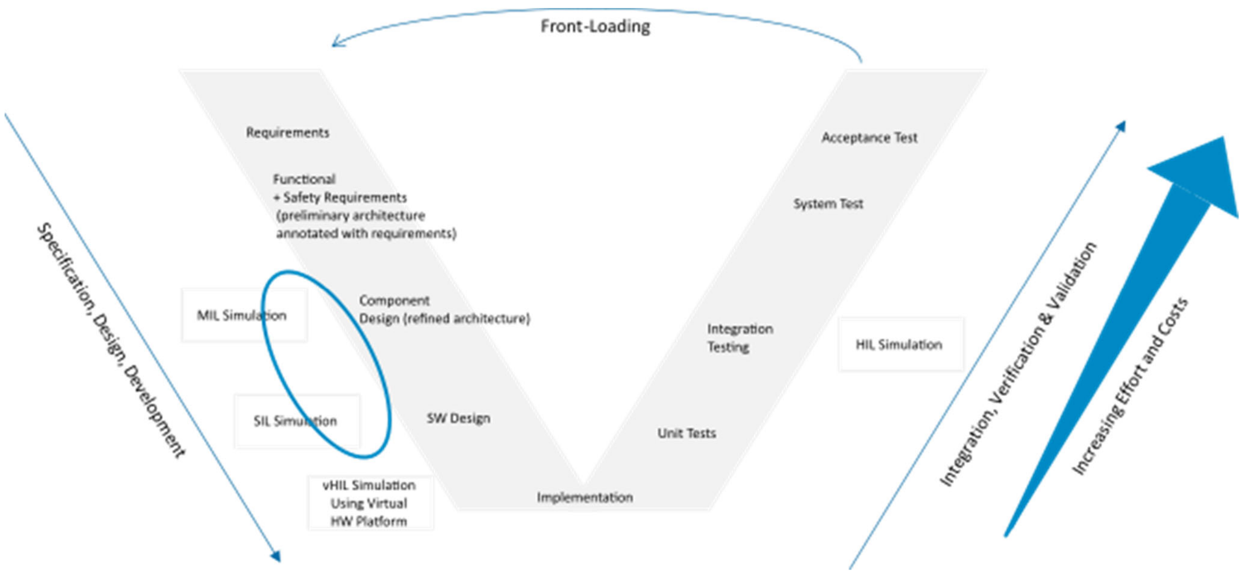


Figure 29: Relation of V&V concept for the maritime advisory framework and Vee-model

Figure 30 illustrates, which validation properties the approach covers. The comparison of Auto-Routes with Reference Routes already allows to assess the aspects of functional correctness, safety and data integrity of the advisory system. In addition, the simulation via HAGGIS allows to assess the aspects of temporal correctness and availability within the virtual test environment.

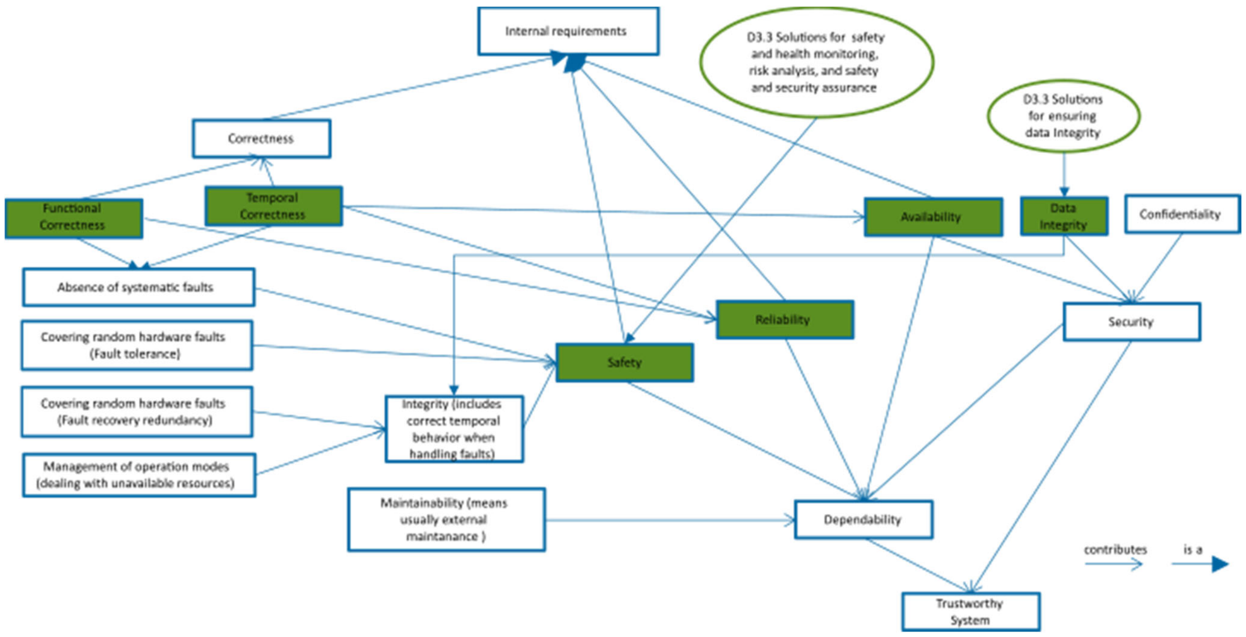

Internal requirements
D3.3 Solutions for safety and health monitoring, risk analysis, and safety and security assurance
D3.3 Solutions for ensuring data Integrity
Correctness
Functional Correctness
Temporal Correctness
Availability
Data Integrity
Confidentiality
Absence of systematic faults
Reliability
Security
Covering random hardware faults (Fault tolerance)
Safety
Covering random hardware faults (Fault recovery redundancy)
Integrity (includes correct temporal behavior when handling faults)
Management of operation modes (dealing with unavailable resources)
Maintainability (means usually external maintanance)
Dependability
contributes
is a
Trustworthy System

Figure 30: Relation of the V&V concept for the maritime advisory framework and the validation properties

4.4 Art2kitekt (a2k)

The art2kitekt (a2k) tool suite is a Model-based Systems Engineering (MBSE) framework developed by ITI that is used to assist in the V&V process. This web-based toolchain provides features to help in the design process of high-integrity systems with real-time constraints (Figure 31). It possesses modelling capabilities that allow the introduction of both, the hardware platform of the target system (including but not limited to distributed systems) as well as the application software, at different granularity levels. Jointly, the engineer also introduces the constraints and the deployment information, i.e., in which processing element each task must be executed. The engineer is then able to apply formal analysis algorithms to these models to evaluate the feasibility/schedulability of the configuration under study (e.g., RTA and CPA) offline.

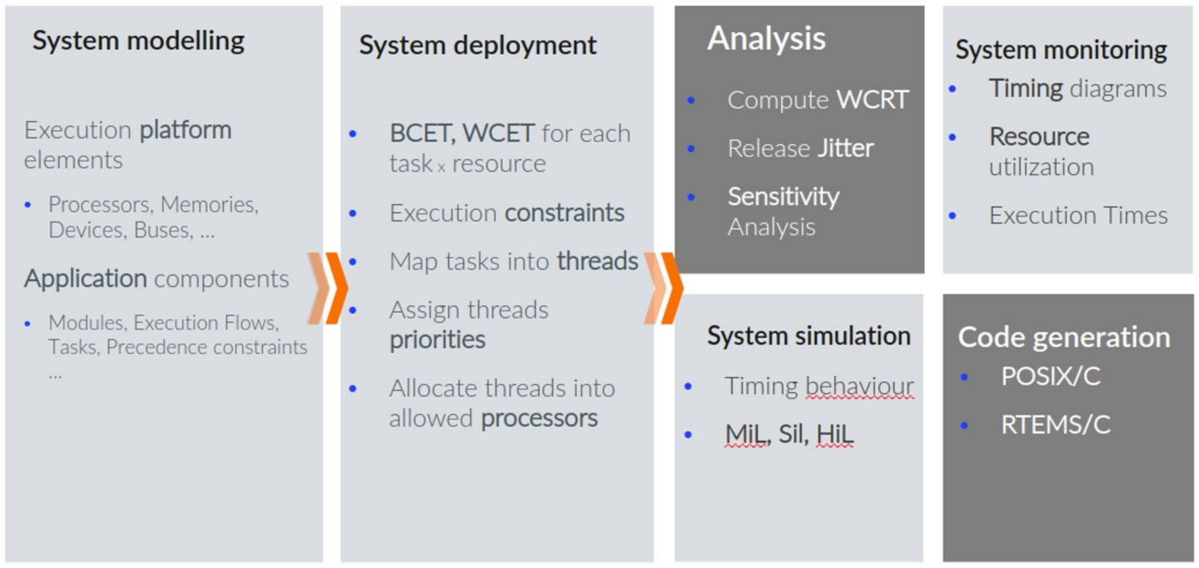


Figure 31: MBSE steps performed by a2k

Schedulability analysis (as the ones previously mentioned) is one of the relevant algorithms to evaluate the feasibility of hard real-time systems. This analysis aims to detect possible deadline misses that could lead to catastrophic consequences. The results obtained through this type of method are generally quite pessimistic, but they ensure that they are reliable. It's the main disadvantage is the significant computing power required to execute it in case the system is too complex. Such systems can be found in the scenarios of the TRANSACT use cases. In those scenarios, for example, a system can traverse different operational modes and the system functionality can be executed in different spots in the device-edge-cloud continuum at run-time. To solve this problem, a complementary approach to this analysis is to simulate the real-time behaviour of the system.

A2k includes a discrete-time event-based simulator to handle these particularly complex scenarios. It supports operational modes and the dynamic behaviour changes that are associated with each change between modes. It also supports events that may change the system in some manner, e.g., the presence of a failure in some component. The simulator computes the response times of the tasks (by utilizing the worst-case response times) and the deadlines that will be missed in the simulated scenario.

Additionally, a2k provides the possibility of generating code templates based on the introduced models. This source code includes the temporal structure of the system, which manages the synchronization and periodicity of the operating system tasks, hence easing the work of the developers. Furthermore, the philosophy a2k works under is that of "correct-by-design". If a model is deemed feasible through analysis, then the resulting code should also be feasible.

Last in the a2k workflow lies the monitoring of the code to verify that the system is behaving as expected during the execution of the actual implementation. By introducing trace points, that are automatically generated in the previous step, it is possible to obtain the necessary information to represent the execution time and response time of the tasks in the operating system. The results obtained can be compared with the initially estimated times in order to bring them closer to reality. Furthermore, observing the system allows for finding spurious or unexpected behaviours not foreseen in an early phase of the design.

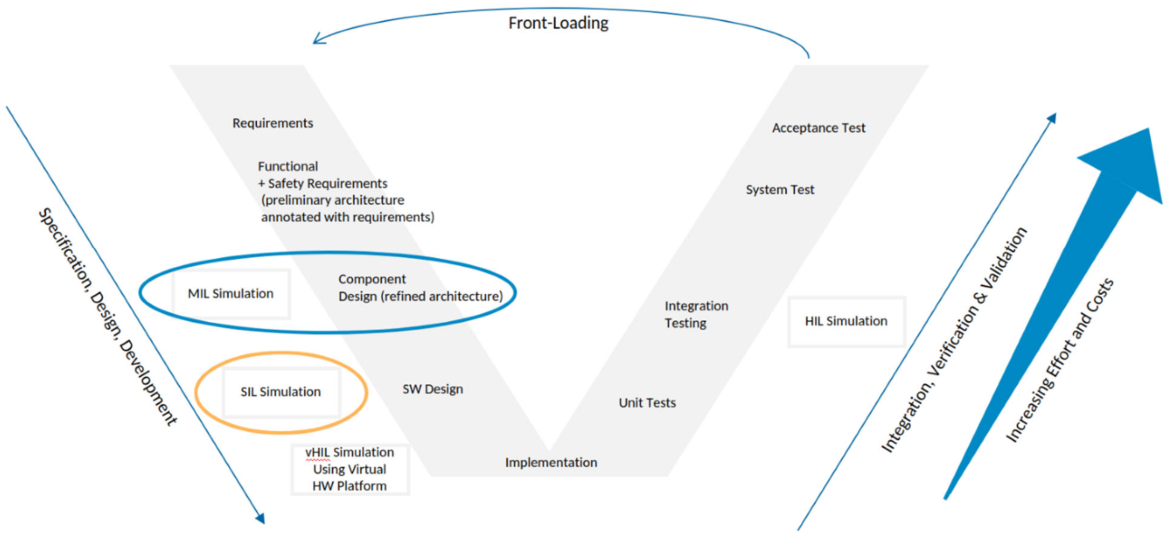


Figure 32: Relation of a2k to V&V Model

Concerning the V&V model (Figure 32), a2k capabilities are mainly focused on the Component Design step. The Model-in-the-Loop simulation refers to the discrete-time event-based simulator provided by a2k, responsible for evaluating the real-time behaviour of the distributed system. In this step, the temporal correctness of the SCDCPS must be ensured by model design, avoiding continuing with the development of an unfeasible system and having to modify the design in later stages, where these errors are more expensive to correct.

Regarding the SW Design step, a2k tool also provides the services to manage Software-in-the-Loop simulations based on the HLA standard, with the ability to build large functional models in a common federation.

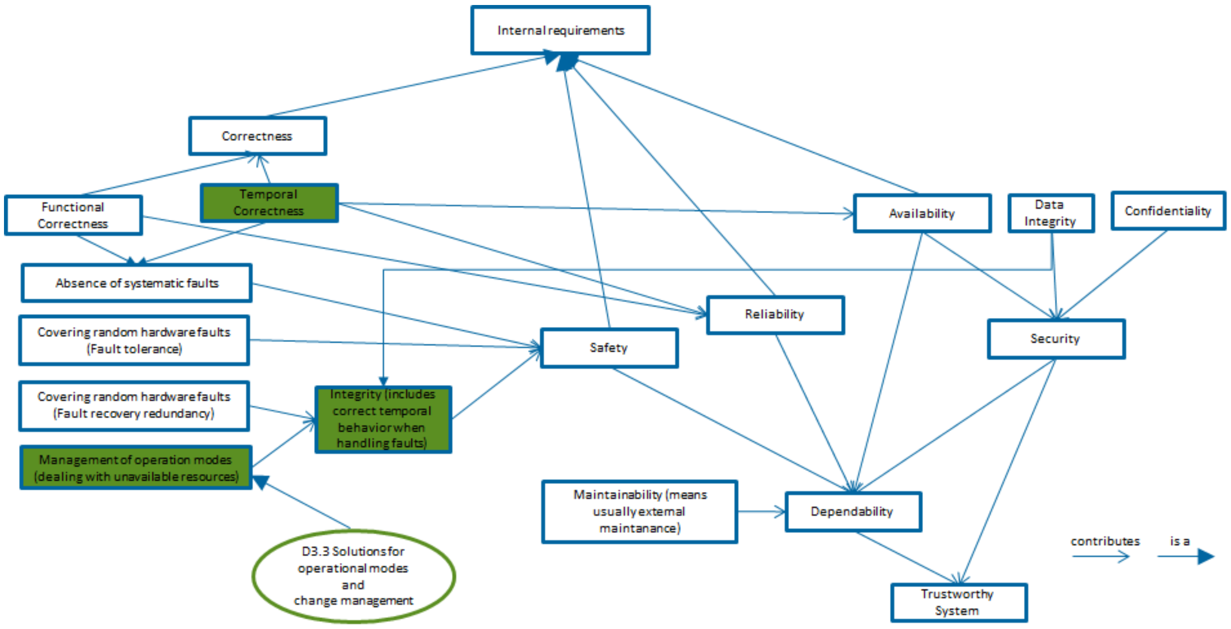

Internal requirements
Correctness
Functional Correctness
Temporal Correctness
Absence of systematic faults
Covering random hardware faults (Fault tolerance)
Covering random hardware faults (Fault recovery redundancy)
Management of operation modes (dealing with unavailable resources)
Integrity (includes correct temporal behavior when handling faults)
D3.3 Solutions for operational modes and change management
Safety
Reliability
Availability
Data Integrity
Confidentiality
Security
Maintainability (means usually external maintanance)
Dependability
Trustworthy System
contributes
is a

Figure 33: Relation of a2k to potential validation properties

In summary, the validation properties that the a2k tool tackles are the **temporal correctness of the system**, the **management of operational modes** and the integrity of the system (Figure 33). This work has been done as part of D3.3 in the **solutions for operational modes and change management, where ITI describes the keys to deploying the Mode change management on the device and the Mode change coordination across the device-edge-cloud** continuum.

4.5 Validation of Image-Guided Therapy (UC4)

In the context of Use Case 4, partners are exploring a cloud-based solution for minimally invasive interventional treatment. Currently, the solution is deployed on a local platform inside the intervention room. The ambition is to migrate these algorithms to the cloud, with the expectation of boosting performance and enabling faster innovation.

Figure 34 Example image of cloud assisted image guided therapy

If multiple hospitals require the cloud services simultaneously, more resources can be allocated in the cloud to cope with this added load. Due to the time varying nature of the request for cloud sources that is associated with multiple hospitals, running multiple operations during the day, dynamic scaling functionality is required. This will ensure that sufficient resources are allocated such that Quality-of-Service demands (e.g. response time) are met.

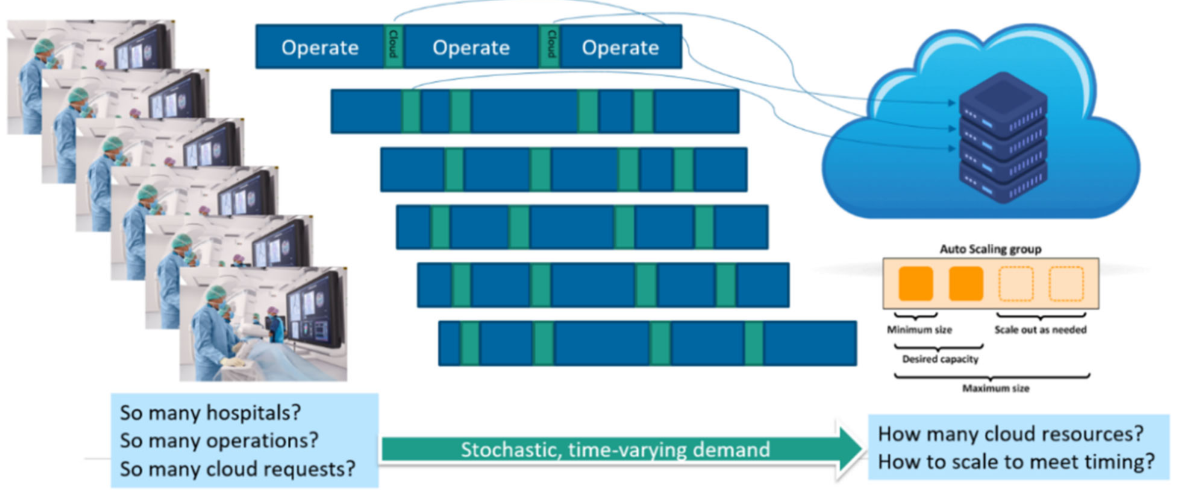


Figure 35: Time varying demands for cloud resources in the case of multiple hospitals, rooms, patients during the day

The interplay between hospitals and cloud compute resources has been simulated using POOSL (Parallel Object-Oriented Specification Language). POOSL supports discrete-event simulation in the early phases of architecture design, thanks to functional mock-up validation and performance analysis. For the Use Case 4 application a simplified model has been created, consisting of 14 hospitals, 5 operating rooms per hospitals and 5 patients per room, distributed during the according to real-world data.

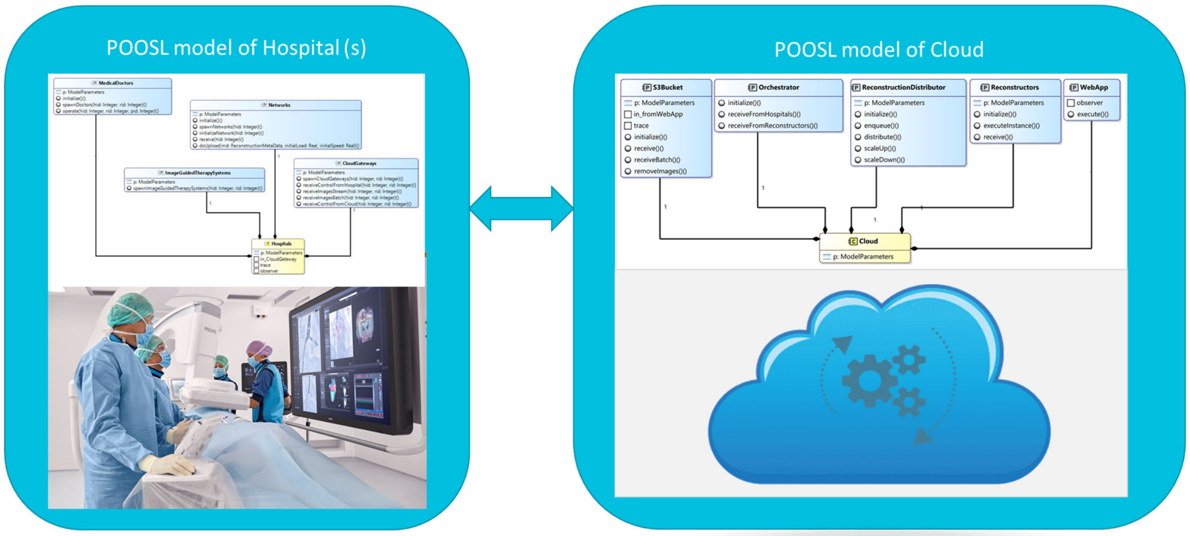

POOSL model of Hospital (s)
MedicalDoctors
p: ModelParameters
initialize()()
spawnDoctors(hid: Integer, rid: Integer)()
operate(hid: Integer, rid: Integer, pid: Integer)()
Networks
p: ModelParameters
initialize()()
spawnNetworks(hid: Integer)()
initializeNetwork(hid: Integer)()
receive(hid: Integer)()
doUpload(md: ReconstructionMetaData, initialLoad: Real, initialSpeed: Real)()
ImageGuidedTherapySystems
p: ModelParameters
spawnImageGuidedTherapySystems(hid: Integer, rid: Integer)()
CloudGateways
p: ModelParameters
spawnCloudGateways(hid: Integer, rid: Integer)()
receiveControlFromHospital(hid: Integer, rid: Integer)()
receiveImagesStream(hid: Integer, rid: Integer)()
receiveImagesBatch(hid: Integer, rid: Integer)()
receiveControlFromCloud(hid: Integer, rid: Integer)()
Hospitals
p: ModelParameters
in_CloudGateway
trace
observer
1
1
1
POOSL model of Cloud
S3Bucket
p: ModelParameters
in_fromWebApp
trace
initialize()()
receive()()
receiveBatch()()
removeImages()()
Orchestrator
initialize()()
receiveFromHospitals()()
receiveFromReconstructors()()
ReconstructionDistributor
p: ModelParameters
initialize()()
enqueue()()
distribute()()
scaleUp()()
scaleDown()()
Reconstructors
p: ModelParameters
initialize()()
executeInstance()()
receive()()
WebApp
observer
execute()()
Cloud
p: ModelParameters
1
1
1

Figure 36: POOSL model of the system including cloud workflow.

The results of the simulations showed that 1 cloud resource instance would lead to unacceptable response times. Introducing dynamic scaling showed that 1-3 cloud resources are necessary and sufficient to meet the response time requirement in all cases.

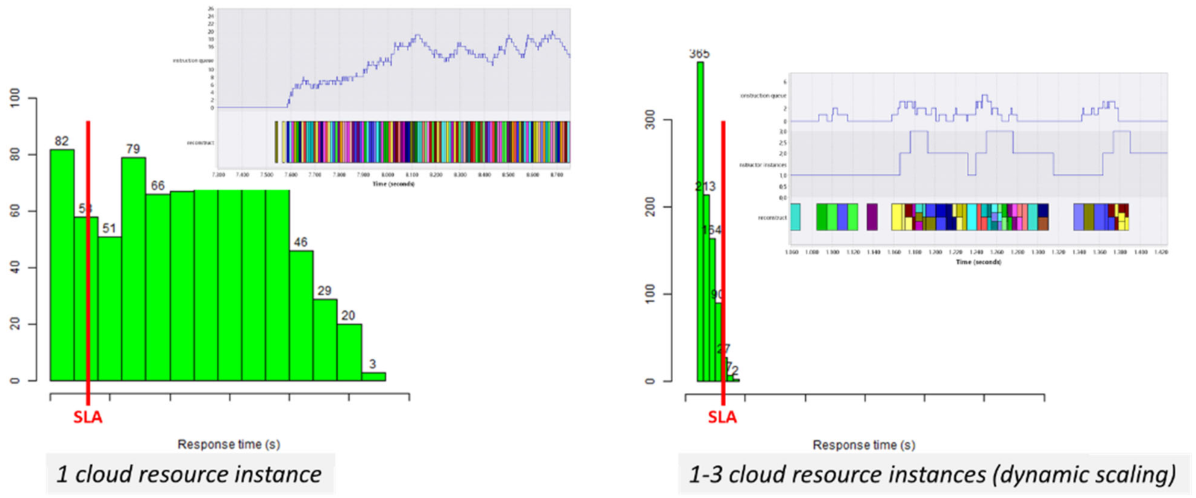


Figure 37: Response times for 1 cloud resource instance (left) vs dynamic scaling with 1-3 cloud resources (right)

The relation to the validation properties is indicated in Figure 38, where the relevant properties are marked in green.

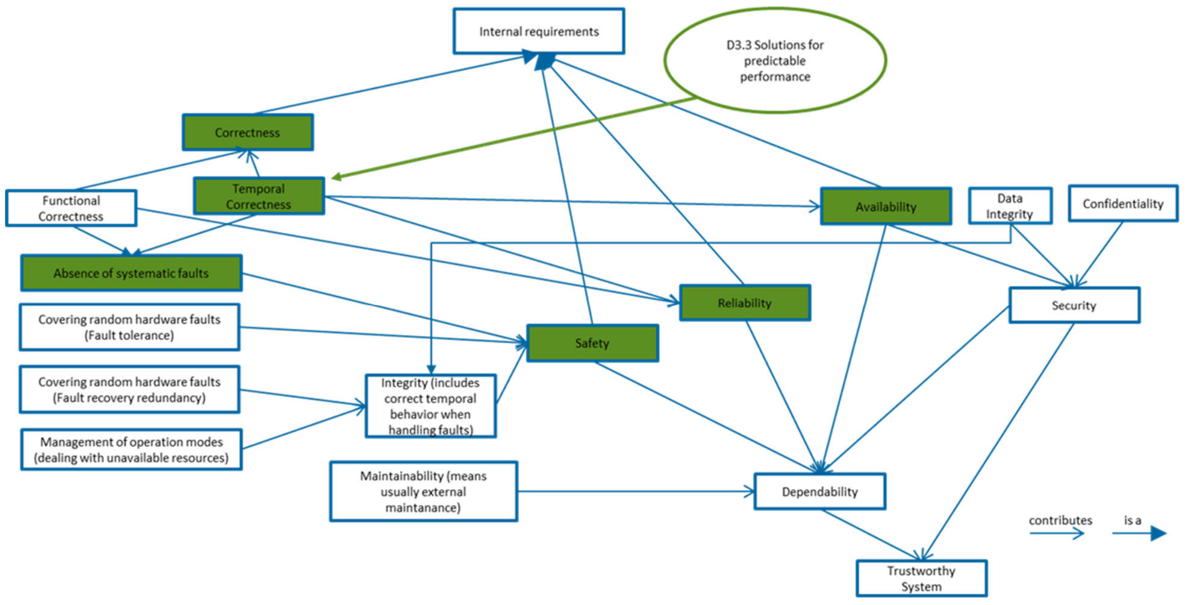

Internal requirements
D3.3 Solutions for predictable performance
Correctness
Functional Correctness
Temporal Correctness
Availability
Data Integrity
Confidentiality
Absence of systematic faults
Reliability
Security
Covering random hardware faults (Fault tolerance)
Safety
Covering random hardware faults (Fault recovery redundancy)
Integrity (includes correct temporal behavior when handling faults)
Management of operation modes (dealing with unavailable resources)
Maintainability (means usually external maintanance)
Dependability
contributes
is a
Trustworthy System

Figure 38 Relation to validation properties for UC4

4.6 Hybrid Simulation based Validation of Remote Driving (UC 1)

4.6.1 Hybrid simulation methodology

The remote driving of autonomous vehicles in urban context is very demanding to validate. This is because the validation cannot be done in the real urban environment (Figure 39, upper part), because it may cause too big risks for safety of other stakeholders on streets, introduce high R&D cost and be impossible in practice. Therefore, the validation needs to be carried out beforehand in some kind of simulation environment (Figure 39, lower part), e.g., from the perspectives of situation awareness, security and safety as well as unexpected changes in the operational environment (objects/people and other vehicles on street etc.) and related systems (e.g. traffic signs/lights etc.) in real-time. It is obvious that in the simulation environment everything is possible, but the challenge is how it matches with the reality on streets.

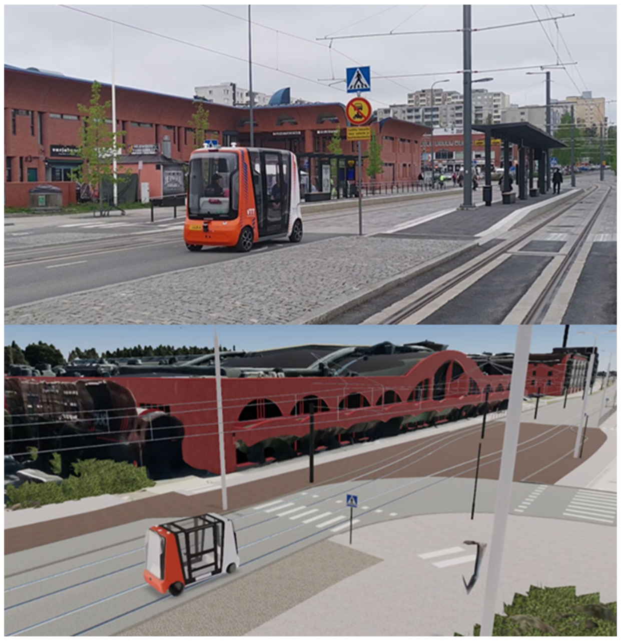


Figure 39. Autonomous vehicle in real world traffic (upper picture), and the respective virtualized autonomous vehicle in digital twin environment (lower picture).

Based on preceding research and experiences, we have here selected a hybrid simulation[3] methodology, where software in the loop (SIL), hardware in the loop (HIL), environment simulations are applied in mixed manner with real objects to reach the validation objective. This methodology can be applied in the UC1 context for example as follows: when validating the interoperability of the digital twin for remote driving and software capabilities of the autonomous vehicle, the hardware of the vehicle and the operational environment of the vehicle need to be simulated. For example, realistic sensor feeds of mobile and other entities in the planned environment need to be fed into the autonomous vehicle SW and digital twin. The benefit for performing the majority of the validation with this kind of hybrid simulation could make the R&D

[3] Ref. Latvakoski, J.; Mäki, K.; Ronkainen, J.; Julku, J.; Koivusaari, J. Simulation-Based Approach for Studying the Balancing of Local Smart Grids with Electric Vehicle Batteries. *Systems* **2015**, *3*, 81-108. http://www.mdpi.com/journal/systems

of the remote driving solutions and validations cheaper and safer. The products are estimated to be more mature, when launching into real urban traffic context.

The relationship between the applied hybrid simulation methodology and the V&V model can be represented, e.g., as depicted in the Figure 41. The validation properties can be related, e.g., to validation of functional correctness, causal temporal correctness as well as absence of systematic faults, reliability, confidentiality, security, and overall trustworthiness of the system as well as some aspects related to safety. If the sensor models would include their behaviour in adverse weather, also the influence of the environmental conditions to the autonomous driving functionality could be included to the simulation studies.

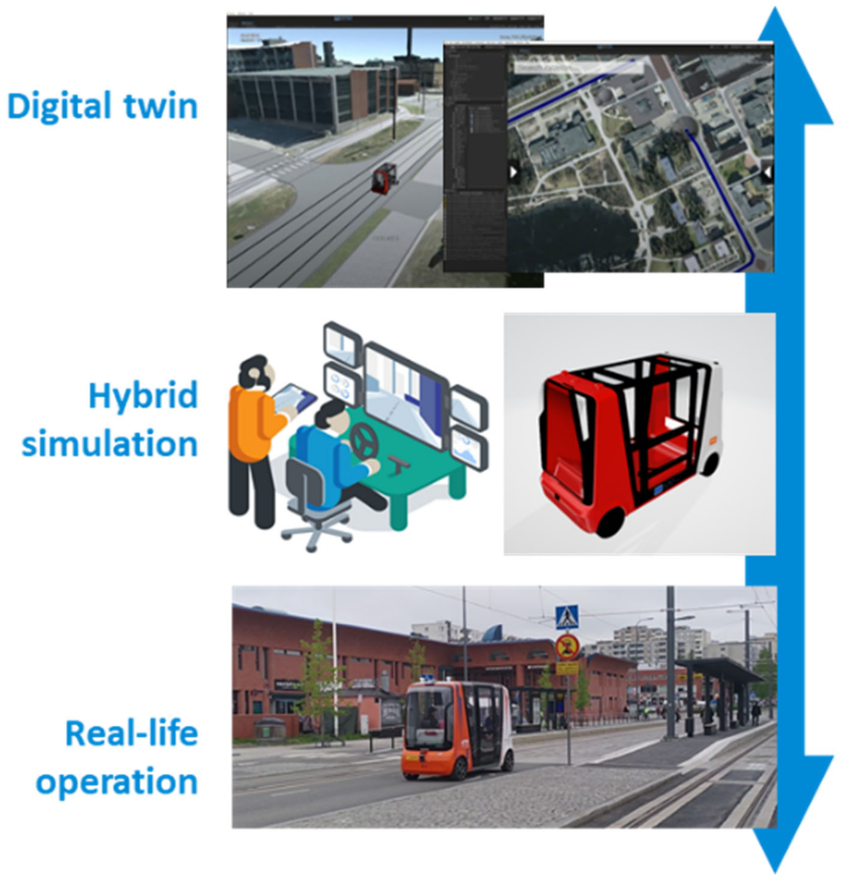


Figure 40. Hybrid simulation approach.

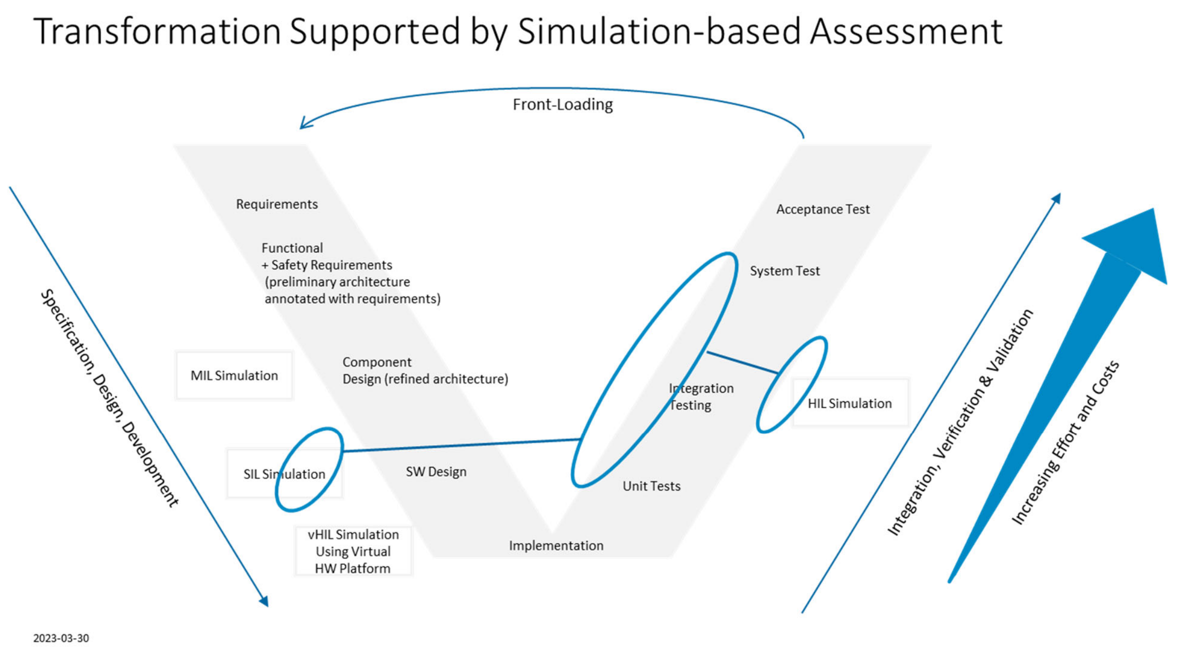

Transformation Supported by Simulation-based Assessment
Front-Loading
Requirements
Functional
+ Safety Requirements
(preliminary architecture
annotated with requirements)
MIL Simulation
Component
Design (refined architecture)
SIL Simulation
SW Design
vHIL Simulation
Using Virtual
HW Platform
Implementation
Unit Tests
Integration
Testing
HIL Simulation
System Test
Acceptance Test
Specification, Design, Development
Integration, Verification & Validation
Increasing Effort and Costs
2023-03-30

Figure 41. Hybrid simulation method as the part of the V&V model simulation-based assessment.

In the following, we first describe the operation of digital twin, then the vehicle simulation, validation of remote driving and finally the validation of the developed security solution in hybrid simulation.

4.6.2 Digital Twin

The digital twin is the software model of the environment where the vehicles will be operating. It's a 3D digital twin of the physical environment that automates generation of context-aware actions on the map, and provides:

- An interactive user interface for humans,
- robust API interfaces for autonomous vehicles
- and integrations for smart city systems and data sources, notably, traffic camera and traffic light integration

The digital twin also enables conducting, planning and controlling cyber-physical interaction, including autonomous vehicles.

4.6.3 Vehicle simulation

Figure 42 depicts the simulation setup used in UC1. Fleetonomy provides a MQTT server through which the vehicle (AUNE) sends and receives messages. AUNE RoutePlanner and PathController softwares manage MQTT messages accordingly. ROS2bridge converts the messages to ros2 messages to be used in SVLSimulator environment. SVLSimulator presents the AUNE movements in Unity -based photorealistic environment.

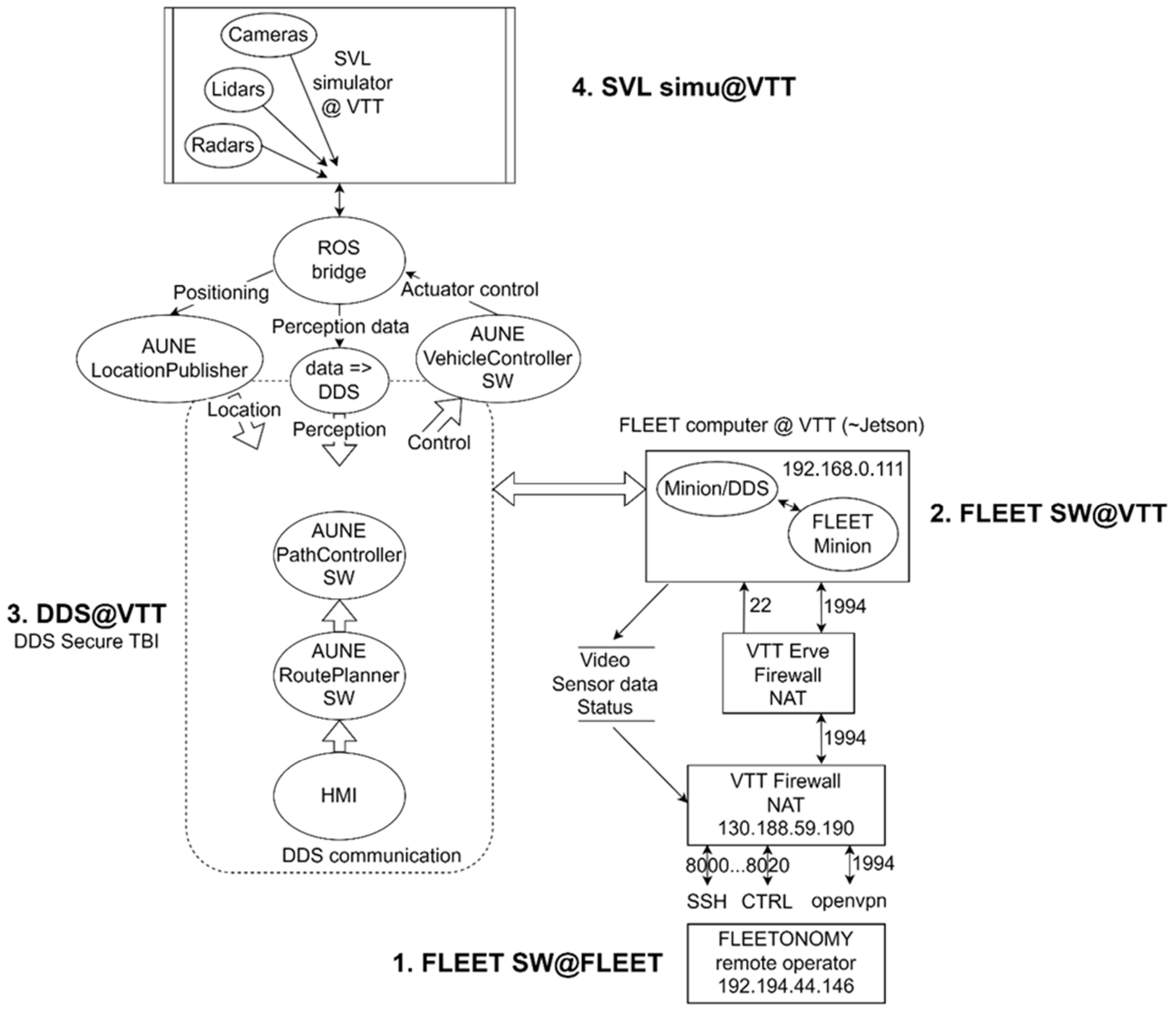


Figure 42: Simulation Setup at VTT

Figure 43 presents the AUNE software stack UI and SVLSimulator at VTT's premises. On the left are all the AUNE's sensors and parameters and on the right the simulated AUNE model is performing the planned task in the precise Unity model of the Hervanta suburb.

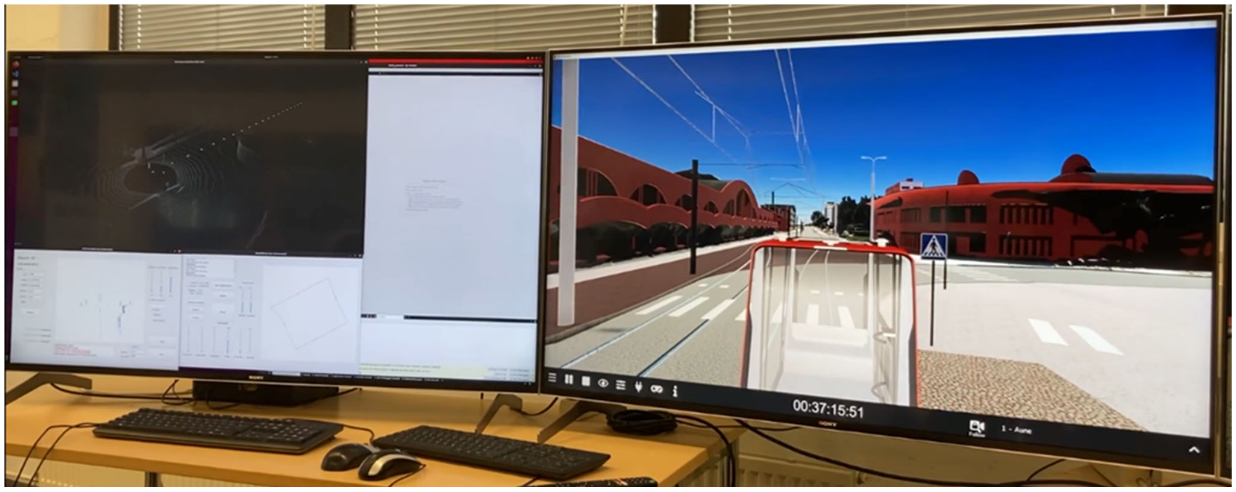


Figure 43: AUNE software stack and SVLSimulator at VTT's control center

4.6.4 Validation of remote driving

Validation is carried out through a combination of tests, including simulations, real vehicle testing, and hybrid testing. The components being tested include:

- Steering
- Throttle
- Brake (different brake types)
- Telemetry
- Other controls (lights, signal, door, etc.)
- Camera streams (quality: glass to glass time, resolution, change of camera)
- State changes (autonomy levels – teleoperation)
- 'preflight' tests; tests run before starting the operation of the vehicle

4.6.5 Validation of digital trust solutions

R&D of the digital trust related solutions[4] (CPSHub Trust@vtt) for the remote driving operation has been challenging because of the lack of other operational components in the development time. The basic structure of the validation system for the trust solutions (CPSHub Trust@vtt) is depicted in the Figure 44. The validation has been/will be carried out by applying hybrid simulation-based methodology in step by step manner as follows:

[4] See D3.4 and D3.6 (draft) section 6.2

- Phase 0: The vehicle and remote driving system are mimicked by using dummy software which provides possibility to use the real implementation of the CPSHub Trust@vtt in similar way as when using simulators or real objects.
- Phase 1: The vehicle and remote driving system are simulated, and the real functional solution of the CPSHub Trust@vtt is validated. The required operation of ground crew member and supervisor with the digital trust solutions is simulated. The status of the operation proceeds and validation results are shown in the dashboard of the CPSHub Trust@vtt.
- Phase 2: The simulated traffic infrastructure system is added into the phase 1 system. The real functional solution of the CPSHub Trust@vtt operation with the three related components with which it needs to operate.
    - After this step, the complex scenarios from the CPSHub Trust@vtt point of view, can be validated by setting the conditions and parameters for the simulated entities so that the wanted impact can be reached.
- Phase 3: The simulated vehicle is replaced by a real vehicle. The remote driving system is still simulated. Especially, the interaction of CPSHub Trust@vtt with the real vehicle is validated.
- Phase 4: The simulated traffic infrastructure system is replaced by a real traffic system. The remote driving system is still simulated. Especially, the interaction of CPSHub Trust@vtt with the real traffic system is validated.
- Phase 5: The simulated remote driving system is replaced by real remote driving system. Especially, the interaction of CPSHub @vtt with the real remote driving system is validated.
- Phase 6: All of the system components are now real. The Security control dialogue with them and CPSHub Trust@vtt can be validated as a real functional operation. When the security control dialogue has been carried out, the real remote driving can start.
- Phase 7: The monitoring of the critical events from the vehicle, traffic infrastructure and remote driving system can now be validated from the real system.

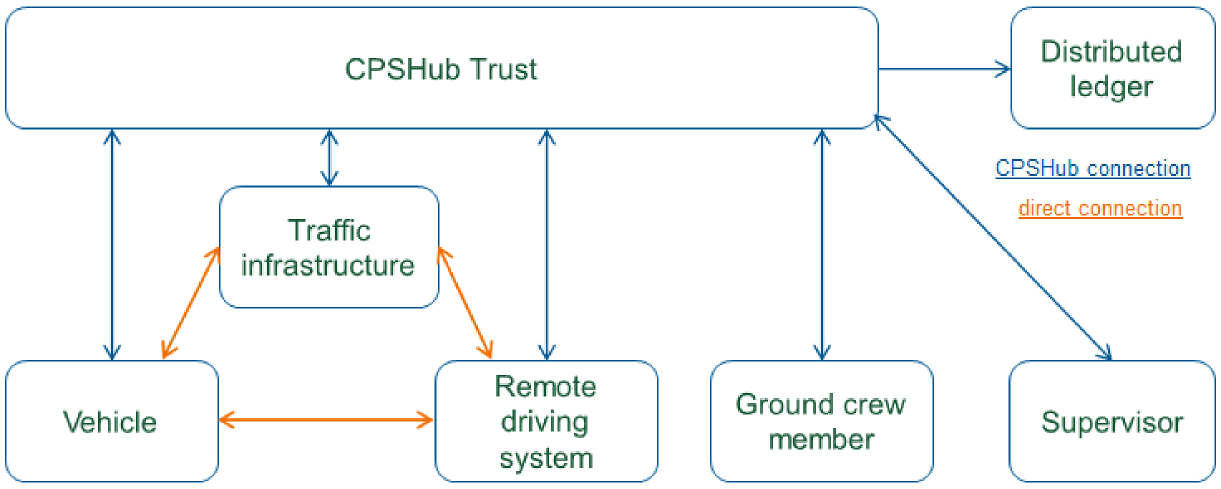


Figure 44. A view to the validation system of the digital trust solutions[5] (CPSHub Trust@vtt).

[5] See D3.4 and D3.6 (draft) section 6.2

4.7 Co-Simulation & Parallel Simulation (UC3)

4.7.1 Overview

In the context of methods and tools for simulation-based design and assessment, Fraunhofer IESE has continued its development of a *Functional Mockup Interface version 3 (FMI3)* standard-compliant integration connector for its simulation and simulator coupling framework FERAL. In the meantime, the maturity of that implementation surpassed the previously supported development level of the still supported FMI2 counterpart. Furthermore, Fraunhofer IESE has implemented an initial prototype of a parallel event *model of computation and communication (MOCC)* which is based on the previously created concept for its realization in FERAL. First evaluation results indicate its beneficial properties in terms of accelerating simulation test case execution in conjunction with co-simulation components, both of which facilitate the design, analysis, and assessment of safety-critical distributed architectures and services as focused by TRANSACT and instantiated in UC3.

4.7.2 Addressed V&V Challenges

Besides the implementation of services for the outsourcing of core functionality from the device tier towards the edge/cloud tier of the TRANSACT reference architecture concept, along with the inclusion of value-added functions for advanced capabilities like AI/ML-based data analytics, auditing, and handling, remote update management of device/edge component-specific software constitutes a significantly challenging feature. In the context of UC3, which tackles cloud-featured battery management for energy-efficient electric vehicles, continuous updating and independent releasing of incremental software versions for each of the potentially high number of vehicles making up a fleet, requires the early and evidence-based verification and validation of a technically sound software distribution strategy. However, this is a tremendously demanding task when considering the different conceivable mechanisms for the systematic delivery of data to potentially mobile target nodes combined with the many situations, where updates might not succeed for a variety of reasons leaving the *safety-critical distributed CPS (SCDCPS)* in an inconsistent state. In view of the huge amount of test cases to be covered by an appropriate testing methodology, any measure other than simulation-based evaluation based on parallelization and co-execution of simulations is insufficient or simply not viable. Therefore, the efficient coupling of simulation models based on divergent levels of abstraction in terms of the implementation detail is paramount, particularly, when being executed atop different virtualization tooling. This, however, also requires an appropriate orchestration of multiple simulation engines based on common data exchange formats as well. Furthermore, comprehensive techniques for the detection and resolution of potential feature interactions must be supported as well.

In view of those challenges, we apply multi-modal simulations, which require the seamless integration of several simulation models to represent the SCDCPS under test with sufficiently authentic fidelity. Therefore, we pre-instantiate the *Virtual Continuous Integration Platform (VCIP)* of the Fraunhofer IESE with TRANSACT-specific terminology and components as described in the next section.

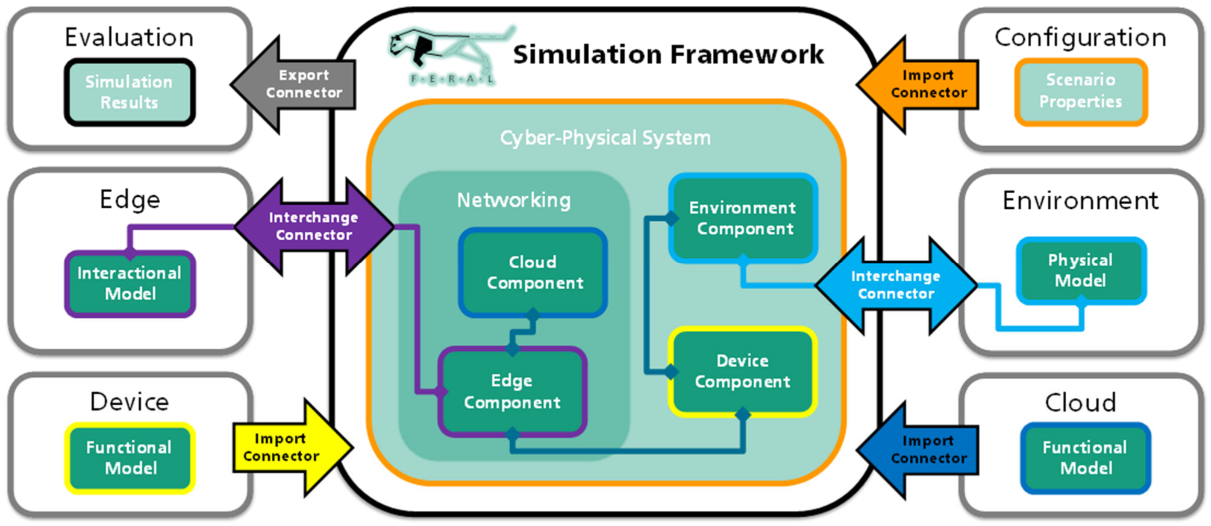


Figure 45: TRANSACT-specific pre-instantiation of the reference architecture behind a VCT-based evaluation setup for the verification and validation of safety-critical distributed cyber-physical systems (SCDCPS) like the cloud-featured battery management system (BMS) of UC3.

4.7.3 Virtual Continuous Testing (VCT) Concept for SCDCPS

Based on the traditional V-Model combined with the shift-left approach, *Virtual Continuous Testing (VCT)* constitutes one of the key activities behind the emerging *Continuous Engineering (CE)* discipline. VCT inherently states that the evaluation of functional and non-functional aspects of every system release candidate shall be performed as early and as often as possible by means of virtual abstraction methods and simulation-based technologies for the sake of sustainable and resource-efficient quality assurance. In this regard, the different phases of the entire development process, from the specification and design to the implementation and deployment, are meant to be executed continuously on an iterative basis, which allows for early prototyping, virtual integration, and incremental testing of each and every expansion stage of the CPS to be developed.

Figure 45 depicts the conceptual pre-instantiation of the reference architecture behind a VCT-based evaluation setup referred to as *Virtual Continuous Integration Platform (VCIP)* developed by the Fraunhofer IESE. It allows for virtually integrating the components of a SCDCPS under development and continuously testing its evolving implementation against existing requirements based on reproducible simulation results. This VCIP setup incorporates all three tiers as defined by the TRANSACT reference architecture, each abstracted by a corresponding simulation component covering Device, Edge, and Cloud Tier representatives. The scenario configuration is complemented by an environment abstraction accounting for all the ambient factors the CPS can be virtually exposed to. Furthermore, diverse alternatives in terms of communication networks might be considered for the coordination and data exchange between each component of an actual SCDCPS, which is eventually deployed in the context of a real-world application domain. To this end, the VCIP instantiation concept anticipates interactional models for most common networking technologies like TCP/IP, Ethernet, CAN, and 5G. After all, any involved functional, interactional, and physical models might potentially be simulated on the same machine or on separate execution platforms while being remotely connected via established co-simulation techniques based on the FMI standards as well as on novel integration solutions like the *FERAL Co-Simulation API (FCAPI)*. With the latter comes the advantage of resource load balancing through distribution across separated computing and storage nodes if required, albeit at the expense of a synchronization overhead, which is specific to the used co-simulation technique and subject to optimization.

4.7.4 VCT-based Verification and Validation of SCDCPS

For the purpose of actually conducting verification and validation of SCDCPS properties, Fraunhofer IESE has further evolved its portfolio of methods and associated tooling. In view of co-simulation technologies, further development steps have been taken to enhance native support for the FMI3 standard by the FERAL-based virtual evaluation platform. Therefore, an improved concept has been created allowing for a facilitated design and implementation of FMI3 connector functionality (see Figure 46), which shall become an integral part of future releases of the FERAL distributable. With the hitherto developed features of the FMI3 connector for FERAL, FMU-based co-simulation scenarios can now be instantiated and executed in analogy to the still supported FMI2-based FERAL connector. Advanced FMI3 features like event-mode are planned to be tackled for the next development increment and will enable further facilities to couple VCT-based evaluation testbed components with external functional logic provided, e.g., as black-boxed FMUs which represent any device, edge or cloud tier service or function. In the context of UC3, this will allow the seamless integration of BMS, Gateway, or Cloud services and functions which need to be mocked or the model (MiL) / software (SiL) / virtual hardware (vHiL) of which needs to be tested in-the-loop depending on the development stage of the SCDCPS as per T&V methodology.

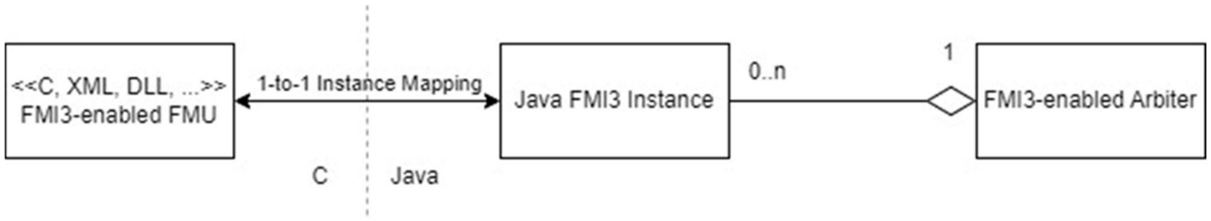


Figure 46: Concept for the support of FMI3 by the FERAL-based virtual evaluation platform.

Regarding the acceleration of simulation-based testing through parallelization, the set of models of computation and communication (MOCC) supported by the FERAL kernel has been extended by a so-called conservative parallel director mode. This mode allows for the concurrent execution of simulation models based on different execution semantics, like discrete event and discrete time, by means of a joint parallel algorithm. Therefore, a Composite Parallel Discrete Event MOCC has been developed constituting the structural root model for the orchestration of different process threads, which, in turn, are meant to execute any contained subordinate MOCCs in a parallel fashion. Subordinate models include Parallel Discrete Event MOCC and Parallel Discrete Event MOCC, both of which rely on the required new implementation of communication links respecting the rules of parallelization, particularly in view of safe time propagation. Figure 47 illustrates the concept behind the parallel execution of simulation models. Initial evaluation trials have shown the effectiveness of this parallelization approach, which can be combined within co-simulation setups in order to profit from both techniques in certain scenarios.

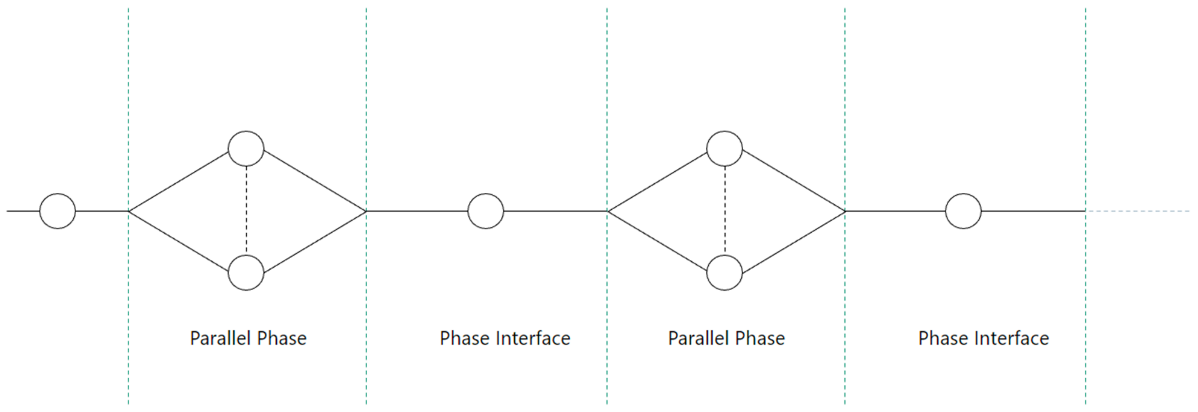


Figure 47: Concept for the parallel execution of subordinate MOCCs as part of the FERAL kernel including alternating parallel phases followed by phase interfaces.

With those two extensions regarding co-simulation and parallel simulation techniques, FERAL-based virtual evaluation testbeds will be able to significantly improve the development process of any complex systems including SCDCPSs which adhere to the TRANSACT reference architecture. In view of UC3, it will be possible to efficiently establish and execute holistic testing scenarios with all relevant system components like BMS, Bluetooth sensors, communication gateway, and cloud-located services and functions being considered concurrently. This will decrease development time and increase the quality of the SCDCPS under test.

4.7.5 Application within System Development

Figure 48 shows the parts of the Vee-model where the evaluation methods based on co-simulation and parallel simulation are applied during the system design and implementation process.

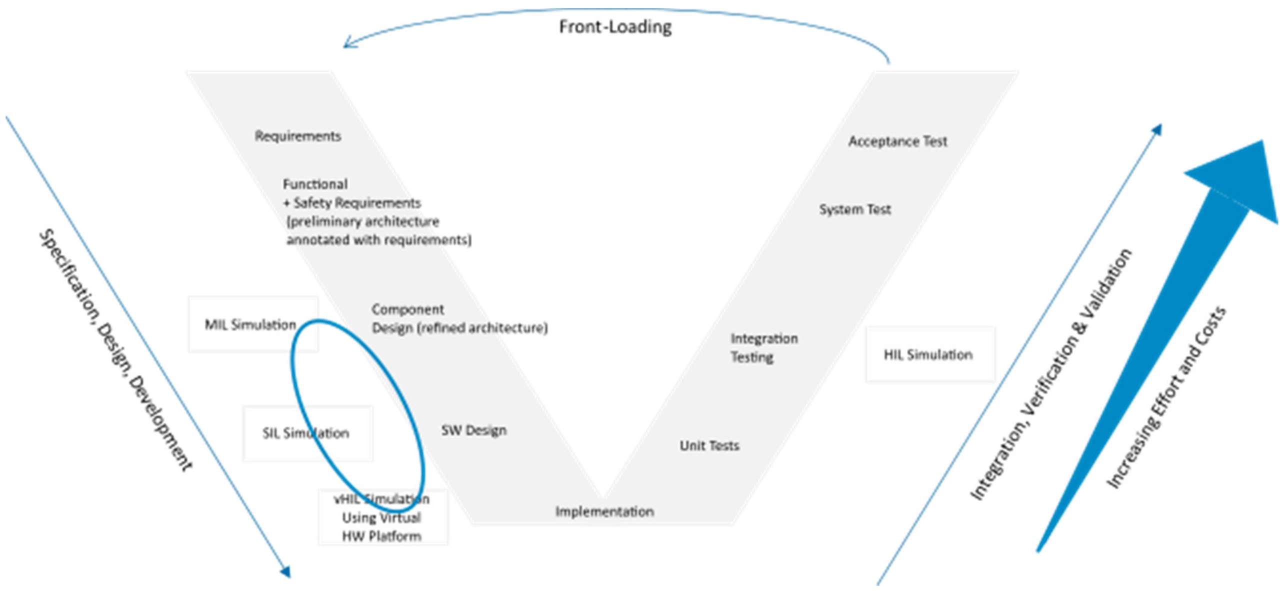


Figure 48: Application of the method during the design process

Figure 49 summarizes which validation properties the VCT-based evaluation approach covers. These include functional and temporal correctness, absence of systematic faults along with Safety and availability in the first place.

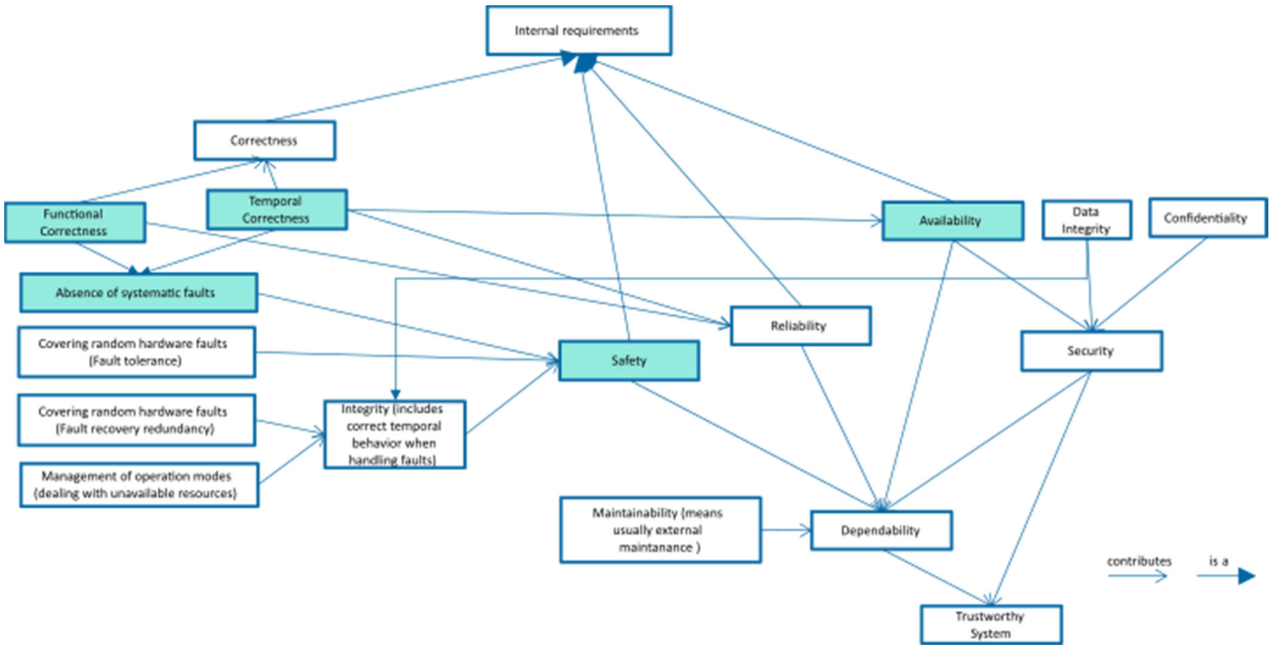

Internal requirements
Correctness
Functional Correctness
Temporal Correctness
Absence of systematic faults
Covering random hardware faults (Fault tolerance)
Covering random hardware faults (Fault recovery redundancy)
Management of operation modes (dealing with unavailable resources)
Integrity (includes correct temporal behavior when handling faults)
Safety
Availability
Data Integrity
Confidentiality
Reliability
Security
Maintainability (means usually external maintanance)
Dependability
Trustworthy System
contributes
is a

Figure 49: Relation to validation properties

4.8 Model Transformations (UC5)

Model Driven Engineering (MDE) is an engineering approach that focuses on the use of models to design, analyse, and build systems. MDE seeks to reduce complexity in the construction stages of systems to speed up development and validation by using different strategies and solutions based on the use of models. For example, model transformation allows deriving concrete implementations from abstract models to automatically obtain solutions as source code or software artefacts. There are three main types of model transformations: model-to-model (m2m) transformations (e.g., to transform a high-level model describing the architecture of a CPS into a low-level model specifying the implementation details), text-to-model (t2m) transformations (e.g., to transform a machine description and characteristics to JSON[6] format), and model-to-text (m2t) transformations (e.g., to produce YAML[7] configurations from a deployment model).

To support the design and validation of the CPS following the TRANSACT architecture, this section proposes an approach based on model transformation for the execution of CPS simulations. Concretely, this approach enables the deployment of the system in a cloud testing environment following the CPS architecture to monitor the infrastructure. Figure 50 shows an overview of the approach and the main components involved.

- The *CPS Architecture Model* addresses the specification of the CPS architecture, including three main concepts: infrastructure, applications, and deployment. The CPS infrastructure involves the specification of sensors, actuators, edge nodes, cloud nodes, and messaging brokers (if the system

[6] JavaScript Object Notation is a text format for data exchange.

[7] YAML is a data serialization format used in this study to represent Kubernetes objects.

implements asynchronous communication). The applications and services that address the system requirements are also specified in the architecture model. These applications correspond to the system functions that address the requirements (i.e., safety-critical, mission-critical, and non-critical functions). Finally, the deployment describes which nodes of the infrastructure will host the functions.

- *Code Generator* is the component that performs the model transformations to generate the simulation deployment and configuration files. This component takes the input model (CPS Architecture Model) and performs a series of M2T transformations to generate several YAML files. The files contain the code for deploying and configuring the functions using a cloud test environment. A monitoring system is also automatically deployed to monitor the infrastructure and quality attributes such as application and node availability.
- Finally, the last stage involves the *System Simulation and Monitoring*, where the generated code is executed to deploy the system (using the cloud test environment) and the monitoring system. A sensor emulator is also run to recreate data generation (using a historical data dataset) from the device tier sensors. The monitoring system provides information about the resource consumption (such as CPU, RAM, and Bandwidth consumption) and the availability of the functions and nodes. For example, it is possible to identify bottlenecks or overloaded nodes that may generate issues.

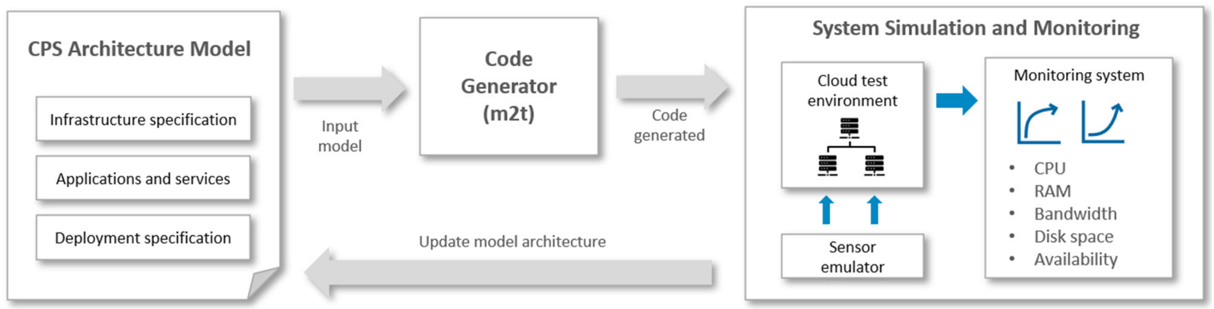


Figure 50: Model transformation approach to support simulation of the CPS architecture

Following the Transact Vee-model (Figure 51), this model transformation approach supports the validation of the TRANSACT architecture implementation through SIL simulations. Using this approach, the architecture implementation is simulated in a Cloud test environment. Additionally, the generation of device layer sensor data is simulated and controlled by means of a *Sensor emulator* (a Python script that can read a historical sensor dataset and emulate the data generation). This enables to evaluate the infrastructure during the execution of the functions deployed on the nodes (edge and cloud) to update and improve the system architecture. For example, If the infrastructure presents problems due to overloads at the edge tier, some non-critical functions could be offloaded to the cloud tier to free resources on the edge nodes and guarantee performance.

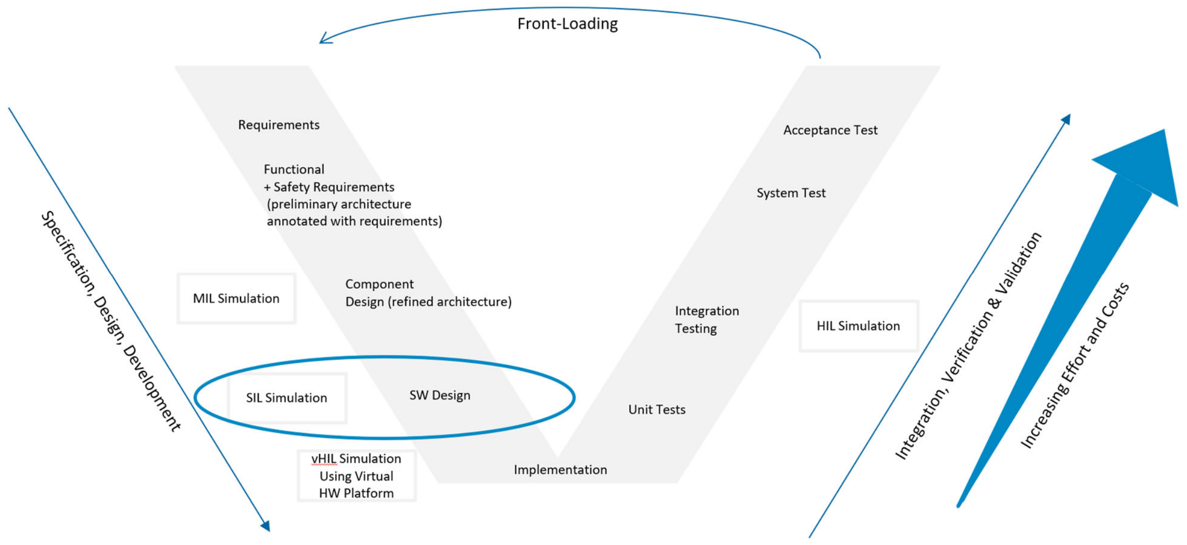


Figure 51: Relation of model transformations in UC5 and Vee-model

In UC5, this approach was designed and focused on the specification of the WWTP CPS architecture, model transformation, and system simulation. The main components developed are as follows.

- A Domain-Specific Language (DSL) for the specification of the CPS architecture. This DSL was developed using MPS[8] providing a mix of editors and multiple notations such as tabular, graphical, and textual.
- A *Code Generator* built in MPS to perform model transformations and generate code. This component generates a group of YAML files for the deployment of the functions (using software containers) and the system monitoring in the cloud testing environment.
- A *Sensor Emulator* to emulate the device tier sensors. This component generates data (like real sensors of the WWTP) reading a historical dataset of five variables: total suspended solids (TSS), chemical oxygen demand (COD), electrical conductivity, pH, and temperature.

Finally, the modelled architecture could be deployed in a test cloud environment by provisioning virtual machines at a cloud service provider. The data generation frequency of the *Sensor Emulator* can be modified to evaluate the capabilities of the infrastructure and functions deployed. Architecture design decisions could be made based on the analysis of the data captured by the monitoring system.

To sum up, the *CPS Architecture Model* is transformed to support the simulation of the system to validate and evaluate the **availability and reliability** (see Figure 52) of the infrastructure and functions of the system. For instance, the TSR40 (the architecture should support the usage of message brokers) is an availability requirement that could be evaluated using this approach as follows. First, to specify the device-edge-cloud continuum architecture model including the functions and message broker(s); then, using the code produced by the Code Generator, deploy the system in the cloud testing environment; Finally, test and analyse the system by modifying the number of clients and the frequency of sending data to the broker(s). The monitoring system collects several resource consumption metrics and displays system information about the

[8] Meta Programming System of JetBrains

availability of nodes and applications (including messaging functions and brokers). If a fault is detected, the architectural model can be modified/upgraded to run the simulation again.

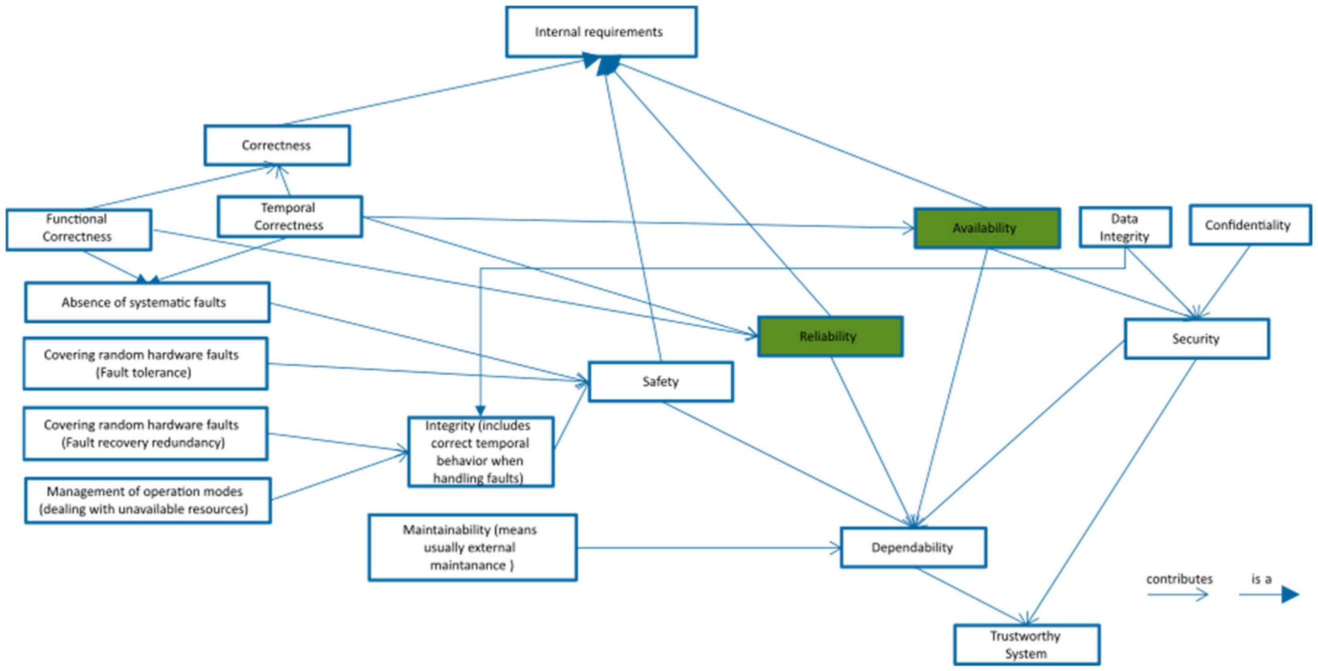

Internal requirements
Correctness
Functional Correctness
Temporal Correctness
Availability
Data Integrity
Confidentiality
Absence of systematic faults
Reliability
Security
Covering random hardware faults (Fault tolerance)
Safety
Covering random hardware faults (Fault recovery redundancy)
Integrity (includes correct temporal behavior when handling faults)
Management of operation modes (dealing with unavailable resources)
Maintainability (means usually external maintanance)
Dependability
contributes
is a
Trustworthy System

Figure 52: Relation of model transformations in UC5 to potential validation properties

5 Conclusions

This document describes the results of T2.2. It explains how the developed methods and tools contribute to the design of a trustworthy distributed safety-critical system and when the methods and tools are used in the development process. Their relationship to the developed solutions of D3.2, D3.3 and D3.4 is presented as well as the relationship of this deliverable to related deliverables.

The methods and tools of T2.2 can speed up the development of SCDCPS *S′* immensely through front-loading of tests. This enables earlier identification of design faults, and it reduces development costs and time to market. The key mechanism used in T2.2 is the investigation of system models defined during the model-based design process of *S′*. The respective analysed system models capture different aspects of the system *S’* and are defined at different levels of abstraction in different phases of the design process. In this way, a landscape of approaches for evaluating different aspects of *S′* during the design is formed, contributing to the development of a trustworthy system *S′* at the end of the design process.

In contrast to the traditional design process, where a system is designed from scratch, TRANSACT transforms an existing CPS *S* into a distributed CPS *S′*. Therefore, many approaches of T2.2 combine models of different levels of abstraction (including real data from existing system parts). Beyond this, the Transition and Validation (T&V) methodology developed in T2.2 describes how the design process can benefit from the knowledge gained from the initial system. It describes how the requirements can be developed iteratively based on the initial system and the intended functionality of the new system. The methods and tools developed in T2.2 can be applied along the development process within this framework.

The developed tools and methods not only cover the different aspects of a trustworthy system that have been derived from the technical requirements of TRANSACT, but also address the challenges arising from the respective use case. Although the development of the approaches in T2.2 was focused on the use case, the approaches address general challenges that arise when developing an SCDCPS, such as evaluating the scalability of a solution, or dealing with the unavailability of resources and ensuring the required performance. The approaches show how such properties can be assessed, and the T&V methodology provides an initial framework for how these different approaches can be applied in a holistic design process where the different aspects of a trustworthy system are established along a step-by-step concretisation process. Necessary changes may need to be pushed back to earlier design stages, but the process guides these activities in a structured and documented way.

6 References

Avizienis, A.; Laprie, J.-C.; Randell, B.; Landwehr, C. (2004): Basic concepts and taxonomy of dependable and secure computing. In: *IEEE Trans.Dependable and Secure Comput.* 1 (1), S. 11–33. DOI: 10.1109/TDSC.2004.2.

Cho, Jin-Hee; Hurley, Patrick M.; Xu, Shouhuai (2016): Metrics and measurement of trustworthy systems. In: MILCOM 2016 - 2016 IEEE Military Communications Conference. MILCOM 2016 - 2016 IEEE Military Communications Conference (MILCOM). Baltimore, MD, USA, 11/1/2016 - 11/3/2016: IEEE, S. 1237–1242.

H. J. Putzer, H. Rues, J. Koch: Trustworthy AI-based Systems with VDE-AR-E 2842-61. Structured development for trustworthy autonomous/cognitive systems. In: EmbeddedWorld2021.

Hasselbring, W.; Reussner, R. (2006): Toward Trustworthy Software Systems. In: *Computer* 39 (4), S. 91–92. DOI: 10.1109/MC.2006.142.

7 Appendix

The following Table gives an overview of the tools and methods developed in T2.2.

Table 6: Overview of Tools & Methods of T2.2

Tool	Purpose	Short Description
Virtual Vehicle (Sect. 4.1)	Verifying and validating the backend architecture by frontloading the tests by using the Virtual Vehicle	The developed tool simulates the whole vehicle and simultaneously utilizes the Telemetric Client Software stack. The Cloud Backend cannot distinguish between real and virtual vehicles.
Risk Storming (Sect. 4.2)	Identify and assess risks aiming at creating a solid and resilient architecture	Method is performed at the design phase at the architecture level by a structured brainstorming approach relying on the experience of software architects, developers and end-users.
Simulation-based V&V concept for Maritime Advisory Service Framework (Sect. 4.3)	Assess risks, efficiency and navigational safety	Reference routes are generated by manually combining various methods and virtual co-simulation is used to examine the Auto-Routes.
Art2kitekt (Sect. 4.4)	Help in the design process of high-integrity systems with real-time constraints	Model-based Systems Engineering (MBSE) framework for modelling, analysis, simulation, development and monitoring of embedded systems with real time constraints in the context of mixed-criticality systems.
Validation of Image-Guided Therapy via POOSL (Sect. 4.5)	Validate the interplay between hospitals and cloud compute resources	The interplay between hospitals and cloud compute resources has been simulated using POOSL. POOSL supports discrete-event simulation in the early phases of architecture design, thanks to functional mock-up validation and performance analysis.
Hybrid Simulation based Validation of Remote Driving (Sect. 4.6)	Validation of remote driving of autonomous vehicle in urban context	Hybrid simulation methodology, where software in the loop (SIL), hardware in the loop (HIL), environment simulations are applied in mixed manner with real objects

Co-Simulation & Parallel Simulation (Sect. 4.7)	Continuously testing of evolving implementation against existing requirements	Multi-modal simulations based on seamless integration of several simulation models representing the system under development/test.