Grant Agreement No.: 101096342
Call: HORIZON-JU-SNS-2022
Topic: HORIZON-JU-SNS-2022-STREAM-B-01-04
Type of action: HORIZON-JU-RIA

HORSE

Holistic, omnipresent, resilient services
for future 6G wireless and computing ecosystems

D5.1 HORSE demonstration scenario

Revision: v.1.0

Work package	WP 5
Task	Task 5.3 Use Cases setup. Demonstration strategy and HORSE platform deployment
Due date	31/03/2024
Submission date	30/04/2024
Deliverable lead	EFACEC
Version	1.0
Authors	Eva Rodriguez Luna (UPC), Jordi Forné (UPC), Jose Manuel Manjón (TID), Gladys Utrera (UPC), Manuel Angel Jimenez (ATOS), Josep Martrat (ATOS), Iulisloi Zacarias (TUBS), Leesa Joyce (HOLO), Pedro Elísio (EFACEC), Paulo Paixão (EFACEC), Stefanos Venios; António Skarmeta (UMU), Juan Tamboleo (UMU), Alexandros Dimos (8BELLS), Alessio Formica (Ericsson), Orazio Toscano (Ericsson), Vito Cianchini (Martel); Alex Carrega (CNIT), Ramin Rabbani (CNIT), Fabrizio Granelli (CNIT)
Reviewers	Xavier Masip Bruin (UPC), Stefanos Venios (S5)

Abstract	D5.1 HORSE Demonstration Scenario is a public document that describes the demonstration strategy to be deployed in HORSE along with the defined use case settings towards a successful evaluation process. The current document is focused in the first HORSE Release but it can be considered as the foundations to a proper evaluation during all the lifecycle of the project
Keywords	HORSE releases; Testbeds; Deployment; Task Forces; HORSE building blocks; Validation; Cyber-attacks; HORSE architecture

DOCUMENT REVISION HISTORY

Version	Date	Description of change	List of contributor(s)
V0.1	15/12/2023	1st version of the template for comments and Table of Contents	ALL
V0.2	04/03/2024	First iteration of the assigned ToC with partners contribution	ALL
V0.3	12/04/2024	Second iteration after Madrid meeting. Task Force 1 and testbed descriptions.	Jose Manuel Manjón (TID), Eva Rodriguez (UPC), Jordi Forné (UPC), Gladys Utrera (UPC), Juan Tamboleo (UMU), Ramin Rabbani (CNIT), Panagiotis Gkonis (NKUA); Alex Carrega (CNIT)
V0.3.1	18/04/2024	Description of Task Force , 2, 3, 4, 6 and technical demonstration strategy	Iulisloi Zacarias (TUBS),Alessio Formica (Ericsson), Rodrigo Diaz, Josep Martrat (Atos), Alex Carrega (CNIT), Stefanos Venios (S5), Alexandros Dimos (8BELLS)
V0.3.2	18/04/2024	Pending TFs. Executive Summary and conclusions, improvements in Use Cases	Manuel Angel Jimenez (ATOS), Vito Cianchini (Martel), Paulo Paixão (EFACEC), Pedro Elísio (EFACEC), Leesa Joyce (HOLO)
V0.4	22/04/2024	Version ready for internal review	Paulo Paixão (EFACEC); Pedro Elísio (EFACEC)
V0.5	23/04/2024	Revision from S5	Stefanos Venios (S5), Orazio Toscano
V0.5.1	24/04/2024	Revision from UPC	Xavi Masip (UPC)
V0.6	25/04/2024	Addressed comments received from internal reviewers (S5, UPC) – First iteration	Eva Rodriguez Luna (UPC); Ramin Rabbani (CNIT); Paulo Paixão (EFACEC), Pedro Elísio (EFACEC), Jose Quesada (Atos); Juan Tamboleo (UMU), António Skarmeta (UMU); Vito Cianchini (Martel)
V0.7	26/04/2024	Addressed comments received from internal reviewers (S5, UPC) – Second iteration	Eva Rodriguez Luna (UPC), Paulo Paixão (EFACEC); Alex Carrega (CNIT)
V0.8	29/04/2024	Version for QA	Paulo Paixão (EFACEC)
V1.0	30/04/2024	Quality assessment and final version to be submitted.	Fabrizio Granelli (CNIT), Josep Martrat (Atos)

Disclaimer

Co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the other granting authorities. Neither the European Union nor the granting authority can be held responsible for them.

Copyright notice

Project co-funded by the European Commission in the Horizon Europe Programme		
Nature of the deliverable:	R	
Dissemination Level		
PU	*Public, fully open, e.g. web*	**x**
SEN	*Sensitive, limited under the conditions of the Grant Agreement*	
Classified R-UE/ EU-R	*EU RESTRICTED under the Commission Decision No2015/ 444*	
Classified C-UE/ EU-C	*EU CONFIDENTIAL under the Commission Decision No2015/ 444*	
Classified S-UE/ EU-S	*EU SECRET under the Commission Decision No2015/ 444*	

* *R: Document, report (excluding the periodic and final reports)*
DEM: Demonstrator, pilot, prototype, plan designs
DEC: Websites, patents filing, press & media actions, videos, etc.
DATA: Data sets, microdata, etc
DMP: Data management plan
ETHICS: Deliverables related to ethics issues.
SECURITY: Deliverables related to security issues
OTHER: Software, technical diagram, algorithms, models, etc.

Executive summary

WP5 and its tasks focus on the HORSE software versions, planned to be released during the lifecycle of the project. This deliverable describes the demonstration scenarios prepared for the deployment, system integration, testing, preliminary demonstration and validation of the achievements of the HORSE project. This deliverable elaborates on the chosen strategy for HORSE development focused on progressively validating the modules being deployed, creating environments for dedicated demonstrations and preparing the testbeds for validation. Also, relevant research has led to selecting and configuring the proper tools to support this methodology.

Right after introducing the rationale for this Deliverable, this document briefly describes the HORSE platform and its architecture, identifying the modules that support key HORSE functionalities as well as the defined requirements. This is the foundation to analyze and to define a strategy to validate the modules and to define demonstrations to assure the HORSE functionalities. The document also describes the collaborative repository developed in HORSE, considering the concept of continuous integration (CI) and continuous development (CD). Moreover, two pilot demonstrations and associated use cases are described.

This document analyses the technical demonstrations and the scenarios to be considered for HORSE validation, describing the proposed Task Forces, emphasizing on HORSE functionalities and on preparing the technical demonstrations around the attack analysis, threat detection, usage of NDTs (including prediction), mitigation actions and enforcement on the infrastructure. The project has focused on three concrete attacks (DDoS for DNS and NTP protocols, DoS signaling in PFCP traffic within cellular core and exposure APIs vulnerability as RNAA) encompassing a set of representative cases that would generalize the HORSE security analysis approach. The project has modelled these attacks and has analysed the impact and elaborated the possible mitigation actions. The document describes the three project testbeds, reflecting the efforts of responsible partners to achieve this goal by creating an agreed common topology (e.g b5G/6G private network) as a baseline to compare results.

Finally, this document presents and describes the roadmap for the HORSE project, highlighting the software versions deliveries and its associated validations campaigns until the end of period 1. It is also enclosed the detailed plan for deploying the first HORSE release and describes the demonstrations for each testbed, illustrating the behavior of the solution in the presence of cyber-attacks.

Therefore, this document can be considered as a foundational reference for a successful evaluation process applicable to both current and future HORSE releases.

Table of Contents

List of figures

List of tables

Abbreviations

3GPP	3rd Generation Partnership Project
5G	Fifth Generation of Wireless Cellular Technology
6G	Sixth Generation of Wireless Cellular Technology
AE	AutoEncoder
AFs	Autonomic Functions
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
BSS	Business Support Systems
CAD	Computer-aided Design
CAS	Compliance Assessment
CISO	Chief Information Security Officer
CRAAX	CRAAX is the Catalan acronym of the Advanced Network Architectures
CSP	Communication Service Providers
CSV	Comma-Separated Values
DDoS	Distributed Denial-of-Service
DEME	Detector and Mitigation Engine
DevOps	Development Operations
DFP	Dynamic Function Placement
DFP	Dynamic Function Placement
DT	Digital Twin
DTE	Distributed Trustable AI Engine
Dx.x	Deliverable x.x
E2E	End-to-end
EM	Early Modelling
ENISA	European Union Agency for Cybersecurity
ePEM	End-to-end (E2E) Secure Connectivity Manager
FL	Federated Learning
GANA	Generic Autonomic Networking Architecture
GUI	Graphical User Interface
HRF	HORSE Reference Framework

HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure-as-a-Service
IBI	Intent-based Interface
IT-1	Iteration 1 of HORSE Architecture Task
IT-2	Iteration 2 of HORSE Architecture Task
IT-X	Iteration X
JSON	JavaScript Object Notation
KPIs	Key Performance Indicators
LCM	Lifecycle Management
LCM	Lifecycle Management
MANO	Management and Orchestration
MEC	Multi-access Edge Computing
ML	Machine Learning
MNO	Mobile Network Operators
MSE	Mean Square Error
NFV	Network Function Virtualization
NFVO	NFV Orchestrator
NTP	Network Time Protocol
OSS	Operations Support System
OSS	Operations Support System
PaaS	Platform as a Service
PAG	Policies and Data Governance
PCAP	Packet Capture
PEM	Threat Detector and Mitigation Engine
PIL	Platform Intelligence
PoC	Proof of Concept
RAN	Radio access networks
RDF	Resource Description Format
REST	Representational State Transfer
RestAPI	Representational State Transfer (API)
RNAA	Resource owner-aware Northbound API Access
RTR	Reliability, Trust and Resilience
SAN	Sandboxing

SDK	Software Development Kit
SDN	Software Defined Network / Networking
SDO	Standard Development Organizations
SM	Smart Monitoring
SMO	Service Management and Orchestration
STIX	Structured Threat Information Expression
STO	AI Secure and Trustable Orchestration
VIM	Virtual Infrastructure Manager
VNF	Virtual Network Function
VSC	Vertical Service Consumers
VSP	Vertical Service Providers
WPx	Work Package x
XR	Extended Reality
YANG	Data Model for Network Topologies

1 Introduction

1.1 Purpose of the document

This document presents the demonstration strategy and describes the provided scenarios to allow the verification and the validation of the HORSE releases, according with the development roadmap of HORSE platform.

1.2 Relationship to other HORSE deliverables

D5.1 is focused on demonstration strategies and demonstration scenarios for the evaluation of HORSE releases, and is related to the Deliverables stated bellow, involving architecture and development:

Deliverable D2.2 – HORSE Architectural Design (IT-1)

Deliverable D3.1 – HORSE Platform Intelligence developed (IT-1)

Deliverable D4.1 – HORSE AI-assisted human-centric Secure and Trustable Orchestration (IT-1)

1.3 Structure of the document

Before diving into the content of this document, it is important to get familiarized with its structure. This section will provide an overview of the different sections that make up the document, easing the navigation throughout the document.

This document is supported by two main sections, structured in the following way:

- Section 2: This section provides a brief description of the HORSE platform including a summary of the primary building blocks. It also describes HORSE's main functionalities and requirements also considering the potential needs defined by the components to be integrated, deployed and tested as well as the functionalities and requirements to validate. In this section the defined Use Cases available at the pilot demonstration stage are revisited accordingly, addressing the requirements related to their setup and deployment.
- Section 3: This section addresses the demonstration strategy describing the defined technical demonstration and the available scenarios for validation mentioning the task forces created to reinforce the development phase. Section 3, also describes the relevant attacks, selected for being used in HORSE demonstration and validation phases. Additionally, this section presents the three testbeds used for HORSE demonstrations, describing the tools used to support the distinct validation stages. Finally, this section presents the planning and the roadmap toward the verification and the validation campaigns, focused on the first HORSE release.

2 HORSE platform

This section provides a description of the HORSE platform for IT-1. It is worth mentioning that a complete description of the architectural building blocks and how they interact with each other can be found in deliverable D2.2 [1]. However, for the sake of completeness, this section includes a summary of the primary building blocks and components involved in the IT-1 HORSE version, as well as their development status and the further actions toward the complete integration into the HORSE platform.

The primary building blocks of the HORSE architecture are: i) the AI Secure and Trustable Orchestration (STO) module, which is responsible for equipping the 6G infrastructure with the necessary performance, reliability, and trust functionalities required for resource orchestration and smart service deployment; ii) the Platform Intelligence (PIL) module, which encompasses a comprehensive range of intelligent strategies and mechanisms that support the predictive approach objective of HORSE and serve as an interface to existing orchestration solutions, as well as; iii) the Intent-based Interface (IBI) module ensures a user-friendly engagement within the overall landscape.

The provision of security and reliability in the HORSE architecture is handled by the STO module. Indeed, the STO module is built upon six components, namely the Smart Monitoring (SM), the Pre-processing module, the Compliance Assessment (CAS), the Reliability, Trust and Resilience (RTR), the End-to-End (E2E) secure connectivity manager (ePEM), and the Domain Orchestrator Connectors.

A brief description of the six STO components is introduced next. The SM module is responsible for the collection of data from various and diverse domain resources, as well as the data involved in the usage of resources in the life cycle management of 6G services. A pre-processing component has been included in the IT-1 version of the HORSE architecture to homogenize the data collected by the SM component. This component is responsible for unifying and standardizing all the collected data. The RTR module provides security and reliability to the HORSE architectural platform. This module orchestrates the generation of mitigation strategies tailored to specific threats, ensuring the precise response to potential security incidents. Additionally, it ensures data anonymization and identity protection for the HORSE end-users. The ePEM is in charge of service orchestration, and feed by the RTR module, supports the recursive deployment of many functional components for multi-tenancy, high device heterogeneity through virtualization, E2E resource self-configuration, and the provisioning of a secure framework that can span across multiple domains and applications. The ePEM will be based upon an open-source MANO orchestrator and it will exploit the domain orchestrator connectors as a southbound interface to control RAN, core, transport, edge, and cloud network segments. Finally, the role of HORSE holds a significant position in the realm of CAS, contributing significantly to the development and implementation of a compliance assessment framework. CAS is responsible for ensuring all enforced security policies and solutions generated by the IBI to meet the considered regulatory framework. Thus, in the case that CAS identifies a non-compliant policy, it initiates a feedback loop for further refinement.

The PIL module is defined in two different contexts; one is a real-time environment, and the other one is an emulated environment. The real-time context is responsible for real-time attack detection and offers mitigation advice based on actual data collection. The emulated context, on the other hand, predicts the attacks and suggests appropriate preventive actions by analyzing the emulated scenarios created by the sandbox. This module consists of intelligent strategies and mechanisms that support HORSE's predictive approaches and serve as an interface for domain orchestration. It comprises of five components, namely the Sandboxing (SAN), the Early Modelling (EM), the Detector and Mitigation Engine (DEME), the Policies and Data Governance (PAG), and the Distributed Trustable AI Engine (DTE).

The SAN module consists of two complementary Digital Twins (DT). The Prediction and Prevention DT that predicts anomalies and threats in the emulated context, and the Impact Analysis DT that determine the impact of applying the mitigation actions or the preventive measures in the different 5G emulated components. Those DTs work together to build a suitable Sandbox to predict and test the modules and topologies of the project. The EM is the module in charge of providing the information to the SAN. This module includes two blocks, the Taxonomy, which characterizes and profiles the different components; and the Attributes, which defines the strategy used to characterize the modules based on the attributes considered. The DEME module works in the "real context" providing threat detection and high-level mitigation advise in the real infrastructure. It analyses and processes network data streams using specific algorithms that offer an in-depth examination of network parameters, protocol headers, and the extensive data collected from network equipment, devices, and Virtual Network Functions (VNFs). The DTE component is feed by the the outcome of the real and emulated contexts, where its Recommender component creates the intent (i.e., a high-level description of the actions to be taken). The Recommender component properly manages the advice generated by both contexts making use of AI models. Finally, the PAG module serves as the comprehensive hub for ensuring data quality, privacy, integrity, and user-friendly access. It facilitates the flow of data while upholding essential legal and ethical data management principles.

The IBI module maps high-level intents into security workflows able to react to security threats and vulnerabilities. It is responsible for aligning the received high-level intents with the configured policies by double checking if the estimated impact is acceptable and if the policies are aligned with the decision to be taken. If these criteria are met, the IBI translates the intent into a security workflow to be taken by the RTR.

The SAN and EM modules are currently utilizing a common topology for the characterization and modeling of attack and have successfully deployed 5G network topology with Kubernetes network emulator (KNE). KNE is an open-source tool that allows the deployment of network topologies on Kubernetes pods. The sub-module DEME is responsible for the detection and reaction of network security threats in the real context. Three different attacks were defined for HORSE, their details are as follows; (i) DDoS attack generated far edge UEs/devices over DNS server element, (ii) DoS signaling attack in the PFCP (Packet Forwarding Control Protocol) traffic between two beyond 5G Core network functions SMF and UPF and (iii) Attack on openness API vulnerability protection – e.g. RNAA attack (Resource owner-aware Northbound API Access), related to the exposing of NEF exposing the 3GPP network capabilities via APIs. The DEME detector provides the information about type of attack detected and the detection confidence value (0-1). The distributed trustable AI engine (DTE) receives the detected outcome from DEME along receives the data in the form of policies from the PAG module via REST APIs. The DTE module collects data from various sources and employs the AI and machine learning (ML) module to define optimal security policies while preserving privacy. The internal part of DTE includes the network data analytics function (NWDAF) aggregator, data processing module, ML model training, ML model evaluator, ML model repository as well as intent creator. ML training is an important module of DTE where various models are trained for the different types of attacks. Four different datasets have been analyzed using ML training and evaluation mechanism.

The IBI module receives the intent (information included by the intent such as threat type, and list of hosts suffering the attack, and the duration of intent remains valid in the network) to the DTE module. It is worth mentioning here that this information is also a part of the automation loop responsible for informing the IBI module whether the selected mitigation action was successful. The RTR module is responsible for applying the prevention or mitigation action to the network elements. One main outcome of RTR is the Ansible playbook that is based on mitigation actions. Lastly, the main functionalities of the PAG module are access management, data anonymization, data retention, data encryption, and observability. The initial release of the PAG component is ready to be integrated as a standalone component with the

aforementioned functionalities, presented with the help of user stories. Moreover, it also includes the storage for storing the meta-data, policies, and demo data store for storing the datasets.

The HORSE project aims to develop new network management strategies while ensuring the user experience with mobility and resource volatility. It focuses on addressing the challenges related to a secure, reliable, and orchestrated continuum in the domain of networks. Numerous advantages can be expected from the HORSE platform. For instance, the project focuses on reducing the time to react to threats or threat predictions, enhancing the incident response processes. The expected benefit of HORSE will be demonstrated and validated during the project lifecycle, in particular with two use cases; SS-LRT (Secure Smart Light Rail Transit) systems and remote rendering to power XR (Extended Reality) industrial case. Moreover, the project benefits include secure distributed operation, fast recovery, timely failure detection, and proactive threat management in smart systems.

2.1 HORSE PLATFORM FUNCTIONALITIES AND MAIN REQUIREMENTS

This section presents the main functionalities and requirements of the HORSE platform and the process of demonstration as well as validation strategy are being designed to assure those features. The complete list of requirements is presented in deliverable D2.1 [2] highlighting the functional requirements and the module responsible in the HORSE framework. Table 1. HORSE main requirementssummarizes the most relevant requirements and the pertinent module(s).

Requirement	Description of Functionality	HORSE Module
REQ-F-02	The HORSE platform must monitor the cybersecurity, including authentication, authorization, threat detection and secure connectivity, of the devices on extreme edge connected to the monitored network	SM
REQ-F-07	The HORSE platform must ensure efficient monitoring mechanisms to timely identify network performance degradation regarding reliability, latency, and bandwidth.	SM
REQ-F-12	The HORSE platform should detect attacks (such as DDoS) on the network.	ePEM, RTR
REQ-F-13	The HORSE platform should propose network and system reconfigurations to mitigate attacks.	DTE, PEM
REQ-F-14	The HORSE platform must implement a methodology to predict an attack.	PEM, EM, DTE

REQ-F-16	The HORSE platform should be able to test and evaluate new configurations before deploying them to the infrastructure.	SAN, EM
REQ-F-19	The HORSE platform must provide the appropriate mechanisms for anomaly detection in the transmitted messages.	PEM
REQ-F-21	HORSE platform must be able to model attacks, and its impact.	EM, PEM
REQ-F-26	The HORSE platform should be able to manage services in a multi-tier environment, comprising cloud, edge and far edge	SM, RTR, ePEM
REQ-F-29	The HORSE platform should provide a way to define policies and action to be performed in the network when some certain conditions are met	IBI, RTR, PEM
REQ-F-32	The HORSE platform should be able to define set of optimum policies using AI/ML models to guarantee the system security against potential attacks	DTE
REQ-F-33	The HORSE Digital Twin should be able to consistently repeat specific experiments, and to incorporate controlled variations to experiments execution as requested by its users.	SAN
REQ-F-34	The HORSE Digital Twin should be capable of deploying different network functions to test them independently or to model whole functionality sets or planes as a single entity, according to specific experiment.	SAN
REQ-F-43	The Horse platform must be able to provide early attack detection within three ROPs.	PEM

Table 1. HORSE main requirements

2.2 HORSE platform deployment

The HORSE platform deployment involves: i) the software source code; ii) the deployment of the software releases in the defined repository, and iii) the deployment of the releases in the demonstration environments. The following sub-sections describe the procedures selected for the HORSE deployment in order to achieve the three steps above.

2.1.1. CI/CD procedures

Continuous Integration /Continuous Delivery-Deployment (CI/CD) is the optimal practice in the development process regarding efficiency, collaboration, and the delivery of high-quality software applications. Continuous integration (CI), as the term suggests, allows for frequent integration and versioning of code, as well as continuous check-ins and code verifications (Zhao et al., 2017 [3] **Errore. L'origine riferimento non è stata trovata.Errore. L'origine riferimento non è stata trovata.**). This strategy will allow for a smooth integration of diverse components that will be successfully incorporated and, most importantly, verified and tested ensuring that new and functional commits will be merged in an already error-free

environment/application. CI succeeds in minimizing the release cycle and allows developers to discover and fix bugs early in time, therefore avoiding backtracking and code validation while providing them with more development and integration time.

Continuous Delivery is strongly connected with CI acting as the next step to the CI/CD pipeline. This stage tries to optimize the infrastructure management and the critical need to balance out time and resources. Validating the software products involves multiple variable inputs like product versions, different operating systems, different third-party software versions etc. It is neither humanly nor practically possible to test all these combinations (Virmani et al., 2015 [4]) However, with the help of automatization, this task will be completed, and at the end of this process, the application is ready for being delivered to the end-users providing “packaging” of the application. Moreover, during this stage tools that are responsible for automated building and releasing of the application are ensured, keeping this way the artefacts always ready for deployment at any given time. Continuous Delivery ensures that the application can be released correctly and more frequently, regardless of code changes which for that fact are kept small-scaled.

Ultimately, Continuous Deployment, the final stage of the CI/CD procedure, provides automated launching and distribution of the applications (and its components). For every change that has passed the previous stages and therefore is able to be incorporated to the end-result, a new deliverable/output/version is released, while in case of a failed test through the pipeline the change is prevented from deployment. This process is fully automated and as such does not require any human intervention. Hence, Continuous Deployment means the ability to bring valuable product features on demand and at will (deployment), in series or patterns with the aim of achieving continuous flow (continuity) and in significantly shorter cycles than traditional lead-times, from a couple of weeks to days or even hours (speed) (Rodriguez et al., 2017 [5]).

2.1.2. CI/CD Components

To satisfy the project’s need for a CI/CD procedure that will guarantee all components in the project to be interconnected and tested, four components are needed:

a. Version Control System (VCS)

   Version control, also known as source control, is the practice of tracking and managing changes to the software code. Version Control Systems (VCS) are software tools that help software teams manage changes to source code over time. VCS keeps track of every modification to the code in a special kind of database. It consists of a remote repository of files that comprise the source code of a software application. If a mistake is made, developers can compare earlier versions of the code to fix the mistake while minimizing disruption to all team members. It is important that this component is integrated with CI tools to monitor the VCS and trigger automated builds, tests and deployments when it detects changes.

b. Continuous Integration / Continuous Delivery (CI/CD)

   Continuous integration (CI) – a primary DevOps practice, allows for automation of the integration of code changes, from multiple contributors to a single software project. Moreover, it allows developers to regularly commit code into a centralized repository where builds and tests then run, thus asserting the new code’s correctness before integration (Atlassian, n.d. [6]). The steps taken in a CI procedure are:

- Developers get copies (locally) of the source code and apply changes to their local system.
- The changes then are committed to the centralized, common repository.
- The server is immediately notified upon any incoming change.
- Finally, the server initiates the below actions:

1. Pulls the latest code version which includes the newly added changes.
2. Builds the application and reports any potential problems.
3. Runs unit and integration tests, reporting any issues if they exist.
4. Releases any artefacts to be deployed for testing.
5. Assigns a build tag to the software version that was built.

c. Test Logs and Report

A mechanism to display results, related to unitary and integration tests, performed after producing testing builds of the software. A build, in this context, refers to the process of compiling the source code, linking it with libraries and dependencies, and generating an executable or deployable artifact. Using this mechanism, the different log files that are generated during program execution and the information about the status of the program, will be shown and will be easily checked. These reports are fundamental in the CI/CD procedure because they help to gain insight about the status and the result of the different tests executed during the development cycle. Moreover, the report generated during the CI/CD procedures is structured in legible files that contain relevant information, such as dependencies related to the software.

d. Container Registry

A stateless, highly scalable central space for storing and distributing container images. They provide secure image management and a fast way to pull and push images with the right permissions. The images that are stored in these containers are preconfigured snapshots of applications and are configured to run in a variety of environments. Container registries are essential in CI/CD because they allow the dev ops team to manage different versions at the same time and in an efficient way.

2.1.3. CI/CD Workflow

Once all the components needed for the deployment of the CI/CD are revisited, a CI/CD scenario composed of different tools is proposed and the suggested workflow is described in

Figure 1 *CI/CD Workflow*and detailed next:

a. The developers commit their code to the project code repository in GitHub.
b. A webhook is triggered (Commit trigger)
c. A Build is prepared in a GitHub Actions Runner
d. Unit testing is displayed directly in GitHub with GA Test Reporter
e. a) If the Build works correctly, it will be published in Docker Hub Registry.
f. b) If the Build or testing fails, the artefact is not ready to publish.

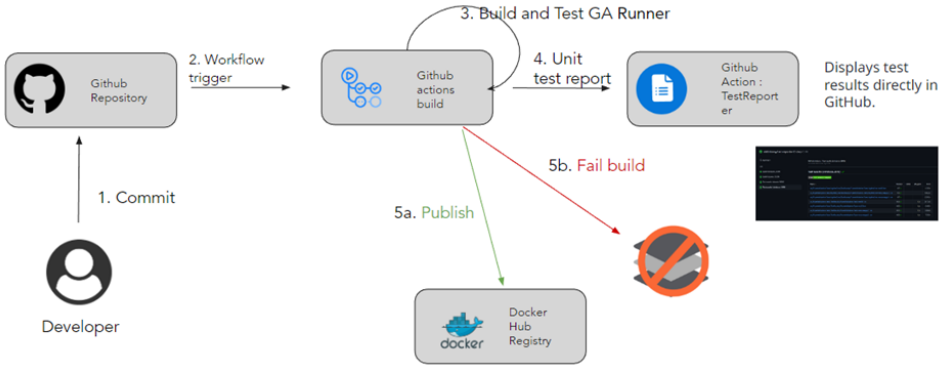


Figure 1 *CI/CD Workflow*

2.1.4. Tools Required for the HORSE Platform Development and Deployment

The following tools (Figure 2) were selected to support HORSE components development and deployment, for the 4 components in the CI/CD process as described in subsection 2.1.2:

- Version Control System: GitHub

  In this scenario, GitHub will be used as VCS. GitHub will fit perfectly into this scenario thanks to its multiple advantages, e.g., security, integration tools and price. Regarding security, GitHub incorporates security features like vulnerability scanning, dependency tracking, and code scanning, helping teams identify and address security issues in their code. Also, GitHub natively integrates with various development tools, CI/CD platforms, and third-party services, offering a wide range of variety. Finally, concerning the cost of this tool, GitHub offers a free plan that allows users access to a variety of resources and functionalities, including unlimited collaborators for public repositories.

- Continuous Integration: GitHub Actions

  GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows the automation of building, testing, and deployment of pipelines. Workflows can be created to build and test every pull request to a repository or deploy merged pull requests to production. Beyond DevOps, GitHub Actions and allows workflows to be triggered when events happen in other repositories. GitHub provides Linux, Windows, and macOS virtual machines to run workflows, or can be self-hosted runners in an own data center or cloud infrastructure. About the pricing, GitHub has no restrictions for free Organizations that have public repositories and brings 2000 free minutes of build time per month. If these limits are exceeded, hosting running is advisable to prevent any decrease in efficiency.

- Logs and Report: Test Reporter (GitHub Actions)

  A GitHub Action that displays test results from testing frameworks directly in GitHub.It is used mainly for:

  - Parsing test results in XML or JSON format to generate a visually appealing report within a GitHub Check Run.
  - Automatically annotating code sections where failures occur, leveraging captured messages and stack traces from test executions.
  - Offering a comprehensive evaluation of test results, including counts for passed, failed, and skipped tests.

- Container Registry: Docker Hub (Public Registry)

  The standard registry for Docker and Kubernetes. It is a highly scalable central space for storing and distributing container images, providing secure image management as well as a fast way to pull and push images with the right permissions and without either administration overhead or resource costs. The only problem with public registries is that we do not have full control over their actions and that they can become expensive if multiple private images are needed.

GitHub	https://github.com
GitHub Actions	https://github.com/features/actions
	https://github.com/marketplace/actions/test-reporter
docker hub	https://hub.docker.com/_/registry

Figure 2 CI/CD Tools

2.1.5. Pipeline Example

Next, the flow associated to configure a basic CI/CD pipeline for testing a Python application with GitHub Actions is shown:

Prerequisites

Ensure the Python project has a requirements.txt file listing all dependencies. The file could look as follows:

```
pytest
```

Structure your project with a src and tests directory. You can execute:

```
mkdir -p src

mkdir -p tests
```

Tests are written in the **tests/** directory, for example, using pytest. Inside the /tests directory must be created a **test_guideline.py** file to illustrate the example with two math tests:

```
# tests/test_guideline.py

import sys

from os.path import abspath, dirname

sys.path.insert(0, dirname(dirname(abspath(__file__))))


from src.guideline import add

def test_addition():

assert add(1, 2) == 3

assert add(-1, 1) == 0

assert add(10, -5) == 5
```

Create GitHub Workflow File

In the root of your Python project directory, must be created a workflow file in **.github/workflows**, such as **python-cd-cd.yml**. Those directories, might needed to be created, using:

```
mkdir -p .github/workflows.
```

Define Workflow

Set up the workflow to run on specific triggers and define the jobs that suit your needs.

```
# Workflow name

name: Python CI/CD Guideline

# Trigger the workflow on push events

on:

push:

  branches:

  - python

# Define the jobs

jobs:

    build:

    runs-on: ubuntu-latest

      # Specify 'python-version' as an output variable to be used by

subsequent jobs in this workflow.

    outputs:

      python-version: 3.9

    steps:
```

```
    - uses: actions/checkout@v3 # Check out the repository's code
    # Set up Python 3.9 environment
    - name: Set up Python 3.9
      uses: actions/setup-python@v3
      with:
        python-version: 3.9 # Specify Python version
```

Job: Run Tests

Set up a job to run the desired tests. Pytest can be used for this purpose.

```
  # Job for running tests
  test:
    needs: build # This job needs to wait for build to complete
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v3
    # Set up Python (version from build job)
    - name: Set up Python
      uses: actions/setup-python@v3
      with:
        python-version: ${{ needs.setup.outputs.pythonversion }} # Python
version based on the output of the build job. This ensures consistency across
different jobs.
      # Install dependencies and run tests
    - name: Install dependencies and run tests
      run: |
        python -m pip install --upgrade pip
        pip install -r requirements.txt
        pip install pytest # Install pytest for running tests
        pytest # Execute tests
```

Job: Deploy Application

Use a placeholder for the deployment stage and replace it with the actual deployment commands when ready.

```
  # Job for deployment
```

```
  deploy:
    needs: test # This job needs to wait for test to complete
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/main' # Only run on the main branch
    steps:
    - name: Deploy
      run: echo "Deploying application..."
      # Add your deployment commands here
```

Push Workflow to GitHub

Commit and push your workflow file and other project files to your GitHub repository.

```
git add .
git commit -m "Add Python project structure and GitHub Actions CI/CD
configuration"
git push origin main
```

Monitoring the Pipeline

Once the workflow file is pushed, GitHub Actions will automatically start the pipeline. The Actions tab in the repository facilitates both workflow progress monitoring as well as troubleshooting any potential failure.

2.3 Use Cases setup

The second validation stage (see section 3.3), estimated to start when the Release IT-2 is available, requires a complete testing environment and will be supported by two demonstration pilots, built upon two well-defined use cases aimed at highlighting and demonstrating the HORSE benefits.

2.3.1 Use case 1: Secure Smart LRT Systems (SS-LRT)

As described in previous deliverables D2.1 [2] and D2.2 [1], the first use case, Secure Smart LRT Systems (SS-LRT), is based on the operation of a Light Rail Transit (LRT) system. A complete operation of a LRT requires a deployment of several IT (Information Technology) equipment (applications and database servers as well as operations workstations) at the Command Center (OCC) area, the deployment of several equipment at the tram stops and requires a network infrastructure allowing the communication between these two environments. For the HORSE demonstration purposes, only a representative scenario of a LRT will be deployed, assuring representative functionalities and the proper behavior of the HORSE modules under validation. Therefore, the preparation of the first Use Case system involves the setup of the IT environment to replicate the OCC, commonly deployed at the Command Center (OCC). Moreover, the setup also includes the network infrastructure

ensuring the connectivity between the OCC and all the tram stops, as well as the configuration of the different devices located at the tram stops.

The required IT infrastructure will be supported by virtual machines, already available at EFACEC premises, and the tram stop devices are also already prepared at EFACEC facilities. Therefore, this virtualized environment can be deployed either at EFACEC laboratory or in the HORSE testbeds (see section 3.1.2 for detailed testbed descriptions and available at M18 of the project). The representative network will be supported by the 5G/6G scenarios, also available in the HORSE testbeds.

HORSE will utilize the setup deployed in this pilot to validate the HORSE outcomes, considering different configurations and scenarios.

Figure 3 shows the overall representation of the SS-LRT Use Case, including all envisioned elements. The SS-LRT Use Case is based in two major Metro operations such as Public Information Systems and Security Systems involving four functionalities:

- Digital signage and public addressing (stations and onboard), that can be affected by ransomware, malware, phishing, data related threats, DoS/DDoS and supply-chain attacks.
- real-time security CCTV systems (stations and onboard) to help dispatchers at operational control centers, drivers, and security agents activities at station's platforms, that can be affected by ransomware, malware, phishing, data related threats, DoS/DDoS and supply-chain attacks.
- Passenger's video help point assistance (stations and onboard) that can be affected by data related threats, DoS/DDoS and supply-chain attacks.
- Security and maintenance agents' awareness (stations and track) that can be affected by data related threats, DoS/DDoS and supply-chain attacks.

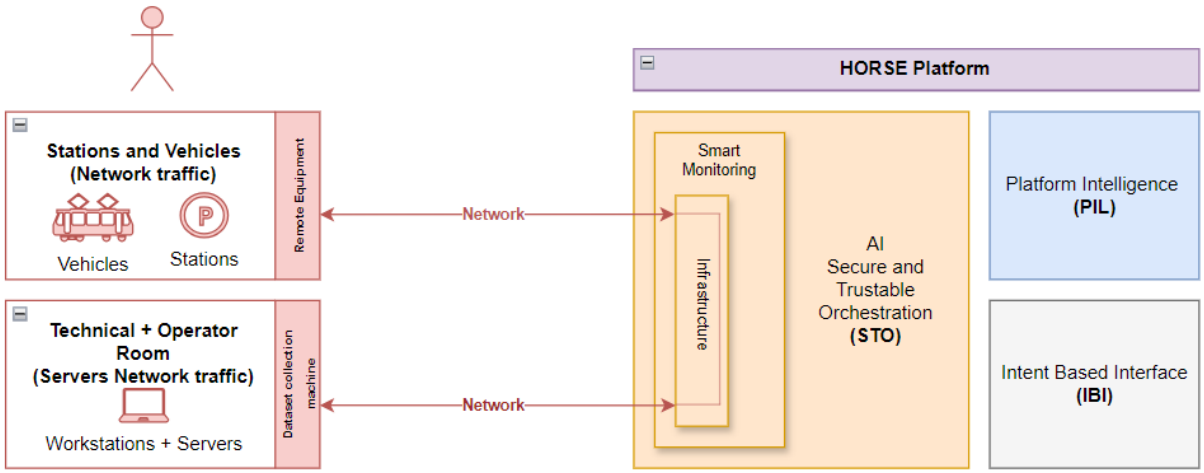


Figure 3 SS-LRT Use Case

Figure 3 also illustrates the main HORSE modules involved in the Use Case, in particular SM, STO, PIL and IBI. Besides these modules also SAN, EM, PEM, DTE must deployed for the SS-LRT setup.

When the IT-2 will be available (see section 3.3) the Use case setup will allow to verify the main benefits of HORSE platform, in particular, allowing to minimize and mitigate cybersecurity threats.

For this validation process, two main approaches can be defined:

Option 1: Hybrid solution using EFACEC laboratories to setup the trams tops and route the information (via internet) to the HORSE testbed where the simulation of the OCC is deployed,

Option 2: All the setup is deployed at the HORSE testbed using the available 5G/6G network for the proper communication between the OCC and the tram stops.

2.3.2 Use case 2 -Remote Rendering to Power XR Industrial (R²2XRI)

The HORSE project's second use case, referred to as R²XRI, is about Remote Rendering for XR Industrial sector. XR technologies are widely used in industries for workflow enhancement. Key benefits include immersive training, remote support, and product design. Industrial XR faces challenges in adapting to technology growth and business demands due to lack of robust, resilient, and secure network infrastructures. The use case utilizes the HORSE platform to address network needs by enabling low-latency and high-throughput data flow. HOLO's XR application Hololight Space (previously called AR 3S), allows multi-player modes for virtual fast-prototyping with CAD files and enables multi-user experiences. Hololight Space includes remote rendering and streaming features from Hololight Stream (previously called ISAR) SDK for high-quality XR experiences. Remote rendering depends on low latency, high throughput, and secure connectivity. The HORSE platform enhances the functionalities of Hololight Space in this use case particularly in the 5G/6G context, validating the components of HORSE platform.

As mentioned in D2.1 [2] and D2.2 [1], this Use Case will address four scenarios:

1. **Rendering of XR in a local network**: Remote application rendering involves a Server Application and a Client Application connected via WebRTC. A stable network connection utilizing HORSE's 5G/6G infrastructure empowers user interaction with the 3D CAD model.

Figure 4 illustrates the remote rendering scenario in R²XRI:

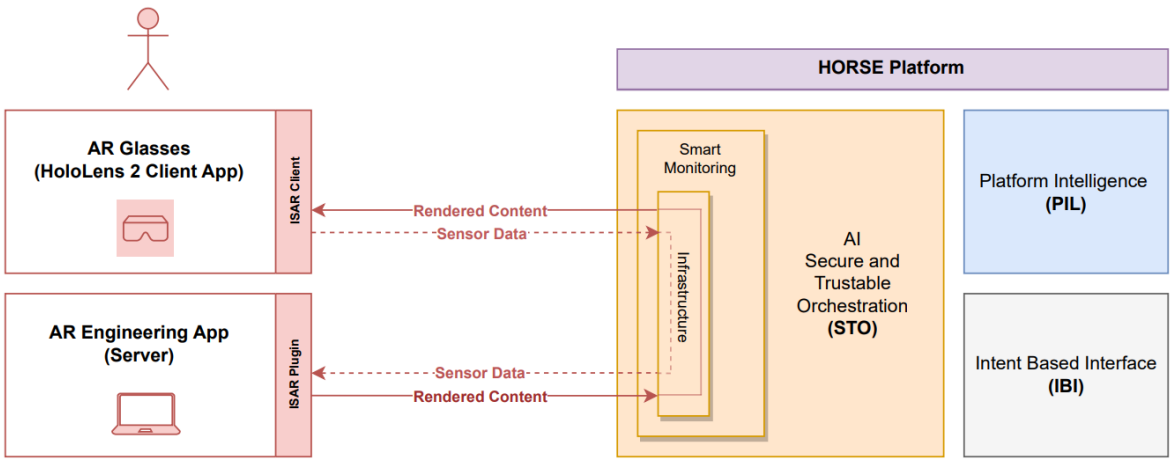


Figure 4 R²XRI Use Case Remote Rendering Scenario

2. **Fast-prototyping sessions in multi-player mode**: Designers and engineers simultaneously collaborate on 3D CAD models on Hololight Space application from different locations. Multi-player sessions utilize the HORSE platform to fulfil the network needs.

3. **Multi-user experience**: Users work independently on rendered CAD files with their own application instances simultaneously in the same network. This setup is common in XR environments. In this use case at least 3 end-users will simultaneously run their sessions on Hololight Space leveraging the HORSE network infrastructure.

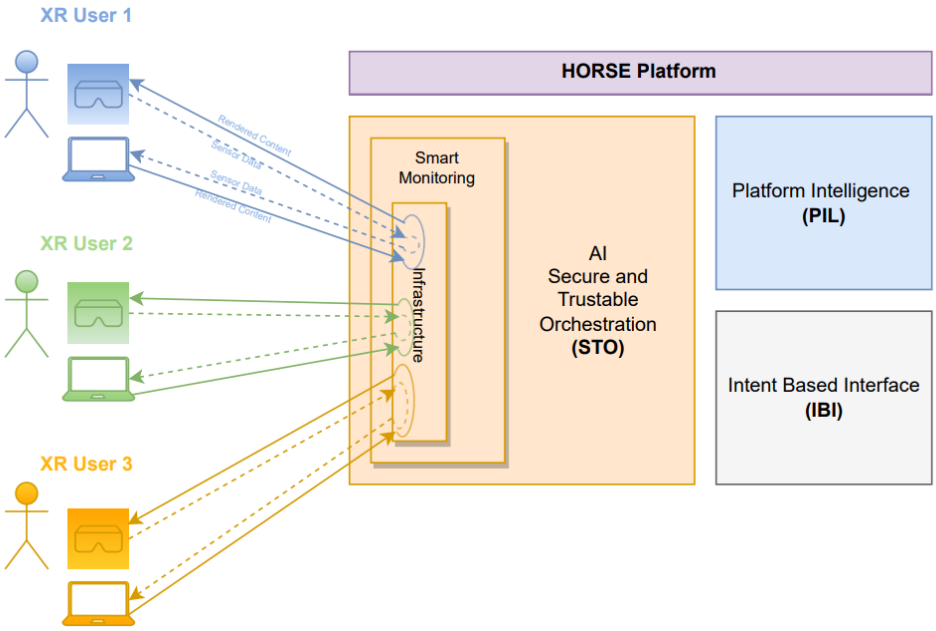


Figure 5 R²XRI Use Case: Multi-user Experience scenario

:

4. **Industrial metaverse and XR devices**: Industrial cooperative tasks in 3D extended reality involve different virtuality levels that reflect the immersive experience. Stakeholders in industrial settings may collaborate using AR and VR for tasks like fast prototyping, factory planning, or training based on user needs. The HORSE platform maintains network infrastructure service requirements such as high-bandwidth, low-latency, and end-to-end security measures for this use case.

In order to validate the HORSE platform in these four scenarios for the R2XRI use case, the following setups can be used.

1. The remote server (e.g., physical workstation, virtual machine) is connected to the 5G/6G network with a cable and the client AR/VR device is connected to the network through the remote server's hotspot.
2. The remote server (e.g., physical workstation, virtual machine) is connected to the 5G/6G network with a cable and the client AR/VR device is connected to the same network through a WiFi capable 5G/6G router.
3. The remote server (e.g., physical workstation, virtual machine) is connected to the 5G/6G network with a cable and the client AR/VR device is connected to the same network through the hotspot shared by a smart phone.

3 Demonstration strategy

3.1 Technical demonstration and validation scenarios

The technical integration and validation of a complex architecture such as HORSE requires an incremental and continuous strategy based on *'divide and conquer' principle* due to the great number of interdependencies. As specified in detailed design documents D3.1 [7] and D4.1 [8], the large number of HORSE modules required a breakdown and module grouping for addressing pre-integrations and technical validation and demonstration. Concretely, it was created six task forces (TF) to address set of modules working together (seeTable 2. This allowed us to progress on technical detail implementation, testing and detailed interface discussion during development phase.

Task Force ID	HORSE modules involved	Partners
TF#1 (WP3)	SAN/NDT, EM and DTE	TID, CNIT, UPC
TF#2 (WP3)	DEME and DTE recomender	ETI, NKUA
TF#3 (WP3)	PAG, PREPROC, database cross	S5, Martel, 8Bells
TF#4 (WP4/5)	IBI, RTR	TUBS, 8Bells
TF#5 (WP4)	ePEM, DOC	CNIT, Atos
TF#6 (WP4)	SM, PREPROC and testbeds	STS, 8Bells

Table 2 Task force enumeration

Moreover, it was decided to focus on three relevant attacks, in order to facilitate developers to think about concrete scenarios where HORSE modules are interacting when facing these attacks in terms of prediction and prevention of security threats. The three considered attacks encompass a variety of well-known representative cases that would generalize the HORSE approach. Additionally, this methodology also facilitates the development of the envisioned attacks modelling, impact analysis and precise mitigation actions as presented in the coming TF sections. The selected attacks are:

1. DDoS attack (Distributed Denial of Service) generated from far edge UEs/devices group over DNS server element. This classic case creates an overall issue on control plane and rerouted targeted network service under attack. As a variant of this sort of DDoS attack, we also consider NTP server case since time synchronization is also a general network service whose attack creates a disturbance in network management.

2. DoS signalling attack in the PFCP traffic between two beyond 5G Core network functions SMF and UPF, impacting on 5G data plane and slicing.

3. Attack on openness API vulnerability protection – e.g. RNAA attack (Resource owner-aware Northbound API Access). This is related to the NF Network Exposure Function (NEF) responsible for exposing the 3GPP network capabilities via APIs.

Finally, it was also defined a baseline network topology to help the interaction between network testbeds and the types of attack, so that we identify the modules affected. The basic topology would correspond to the roll-out of a 5G SA private network (NPN) composed by just one gNB radio, a distributed UPF module deployed at the Edge. DNS and NTP services were also deployed in an Edge node separated from a central DC in which 5Gcore functions are deployed, as depicted in Figure 7 (next section). This is the baseline topology that will be implemented for the three available project testbeds.

3.1.1 Task Force Descriptions

3.1.1.1 Task Force #1

The main module of this Task Force, illustrated in Figure 6, is the Sandboxing module: Prediction & Prevention and Impact Analysis Digital Twins. Jointly with this module, it was explored the data exchange between the Early Modelling (EM) and the Sandboxing.

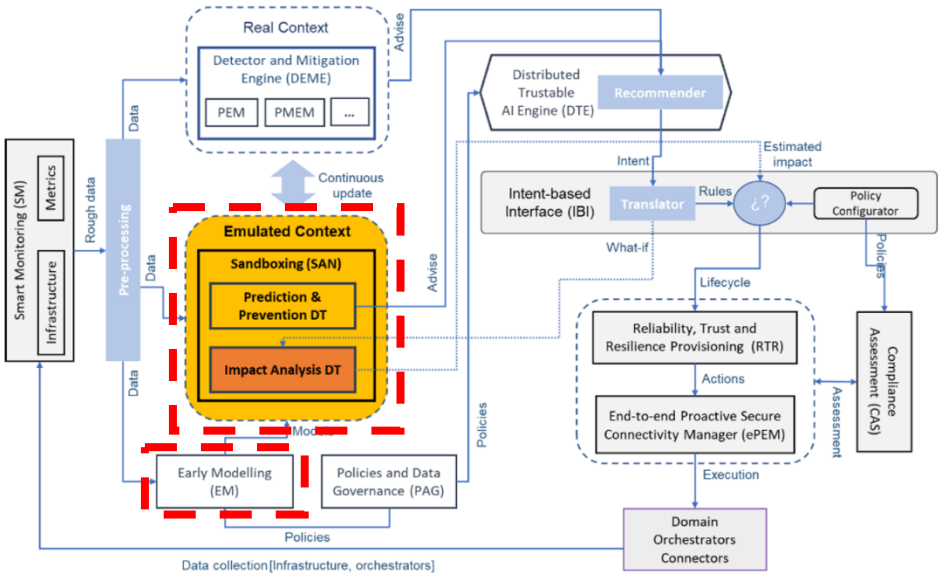


Figure 6 HORSE detailed architecture focused on the Sandboxing and Early Modelling.

First, it was agreed on a common and preliminary topology, to be deployed on both Digital Twins and able to execute a first attack: the DDoS amplification attack.

This topology, represented in Figure 7, is made up of 2 routers, 10 DNS clients, a DNS server and a 5G Core. The descriptor of the topology is based on a YAML file, that contains the images of each one of the modules and the links between them.

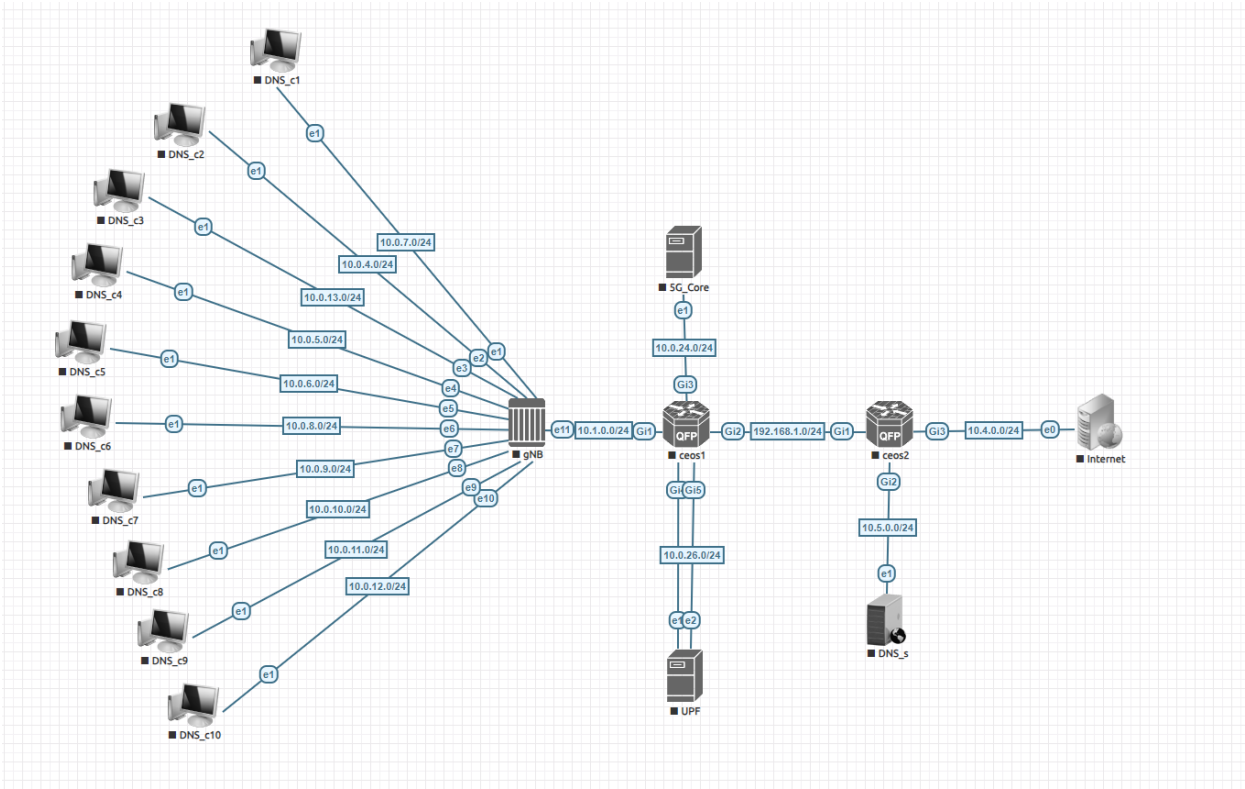


Figure 7 Preliminary topology modelled on EVE-NG.

The DDoS Amplification attack is based on a modified version of DNS-Flood tool codename Thanos, available in a GitHub repository[1].

The YAML topology file and attack descriptor are provided as an input to the Sandboxing module, which contains the Impact Analysis and Prediction and Prevention Digital Twins.

The deployment of the Impact Analysis Digital Twin has been done using KNE (Kubernetes Network Emulator), where the modules are deployed as a Kubernetes pod. Also, a Prometheus installation has been done to monitor this Kubernetes environment.

The deployment of the Prediction and Prevention Digital Twin is based on the Comnetsemu network emulator[2], which allows to deploy containers from real services on an emulated network (capable of implementing any topology and of controlling bandwidth, loss and delay of the links). Comnetsemu is extended to deploy a 5G network (RAN + core) based on UERANSIM (Open Source 5G UE and RAN simulator) and Open5GS (Open Source implementation for 5G Core and EPC)

The work on the Early Modelling has been focused on the definition of the Threat Models for the three attacks that will be considered in HORSE:

- DNS amplification
- DDoS signaling attack from SMF/UPF
- API vulnerability.

A layered approach has been used to identify the type of attack, the impact of the attack, the affected layer, and the component affected by the specified attack. Moreover, the threat model

[1] https://github.com/maxng07/dns-flood/tree/master

[2] https://github.com/stevelorenz/comnetsemu

also identified the entry points that the attacker can choose to launch the attack. Threats have been represented through an XML schema, as well as the appropriate mitigation actions to be taken whenever a threat or a security risk is faced. The attack information has been illustrated using the modeling process proposed to provide an overview of the attack and available countermeasures by linking it with publicly available frameworks like MITRE ATT & CK, MITRE ATLAS, and 3GPP CAPIF.

DNS amplification

The threat modeling information considered for the DNS amplification attack is presented in Table 3 DNS amplification threat modelling informationTable 3 and its XML representation generated by the Early Modeling is presented in Figure 8 DNS Amplification attack XML representationFigure 8.

Attack Category	Denial-of-service
Attack Type	DNS Amplification
Traffic Protocol	DNS, NTP
Impact	Services unavailable
Mitigation Approach	Filter network traffic
Affected Components	SDN, NFV, RAN, Cloud and network services
Affected Layer	Radio, Core network
Entry points	SDN controller, AMF, DNS Server, network functions
Framework	MITRE ATT &CK

Table 3 DNS amplification threat modelling information

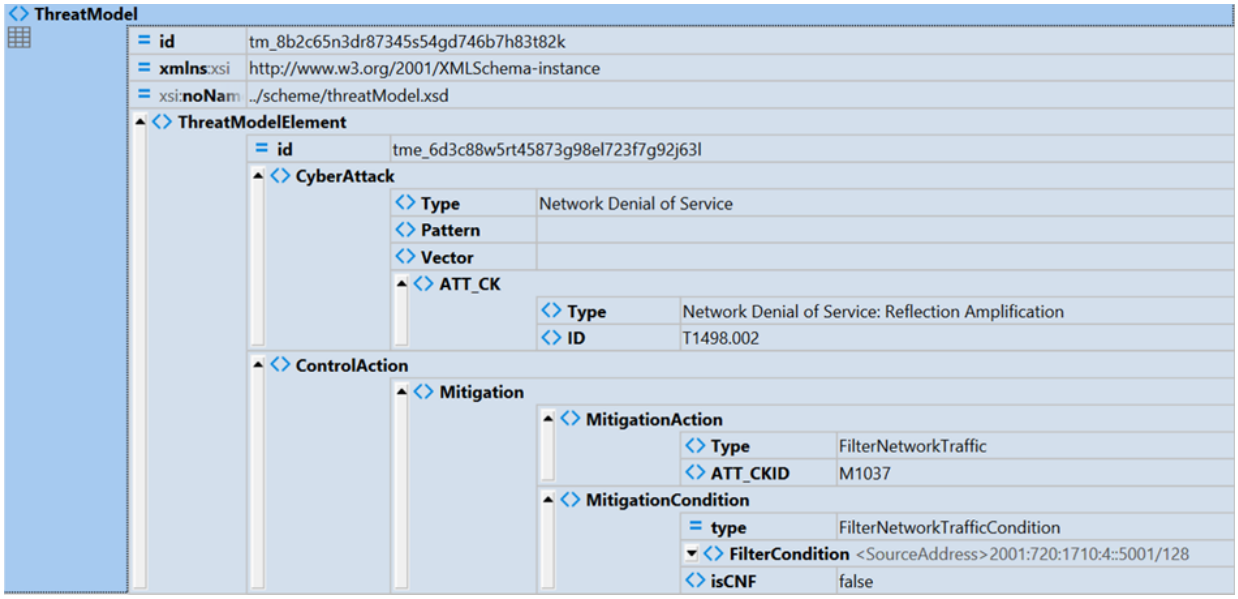


Figure 8 DNS Amplification attack XML representation

DDoS signaling attack

The threat modeling information considered for the DDoS signaling attack is presented in Table 4 and its XML representation generated by the Early Modeling is presented in Figure 9

Attack Category	Signaling threat		
Attack Type	DDoS signaling		
Traffic Protocol	HTTP/2, PFCP	RRC	RRC, SIP
Impact	Unavailability, CN reach to overloaded state, service disruption		
ML model	Naïve Bayesian, SVM, Random Forest	Randomization	CNN
Affected Components	Network services, core network function, signaling server		
Affected Layer	Core Network	RAN	Complex analysis
Entry points	Server, network functions	Server, network functions	

Framework	ATLAS MITRE

Table 4 DDoS signaling threat modelling information.

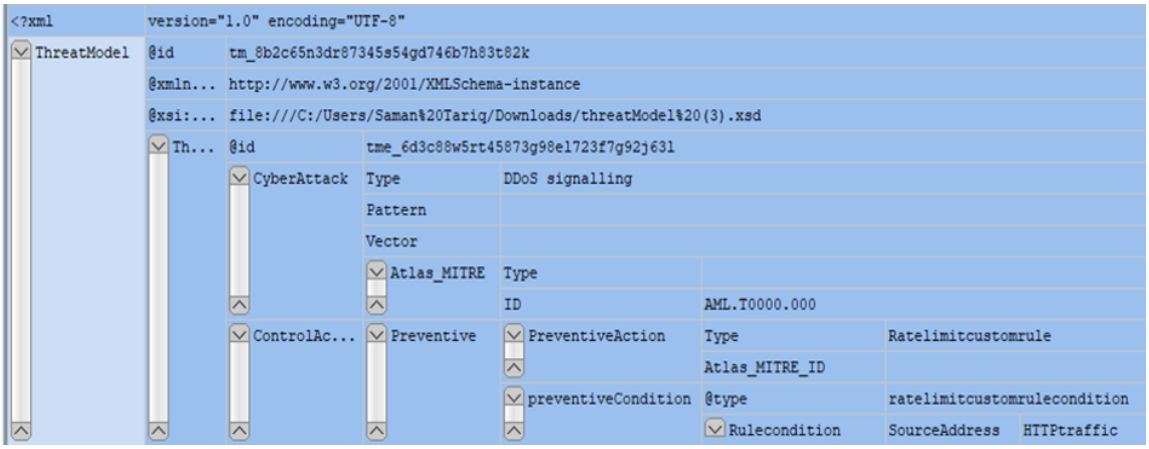


Figure 9 DDoS signaling attack XML representation.

API Vulnerability

The threat modeling information considered for the API Vulnerability attack is presented in Table 5and its XML representation generated by the Early Modeling is presented in Figure 10.

Attack Category	API vulnerability (RNAA attack)
Attack Type	NEF Northbound API
Traffic Protocol	HTTP, Restful API, etc.
Impact	Unavailability of services, confidentiality, Integrity
Mitigation Approach	**Authentication & Authorization** - API instance authorized to the specific end-user network - Enhanced security protocol features - Such as Oauth2 or Mutual TLS - Role-based access control - Limiting access
Affected Components	Core network functions (AMF, SMF, UPF, PCF)
Affected Layer	Service layer

Entry points	External application, API gateway Server, network functions
Framework	3GPP CAPIF (Common API Framework)

Table 5 API Vulnerability threat modelling information

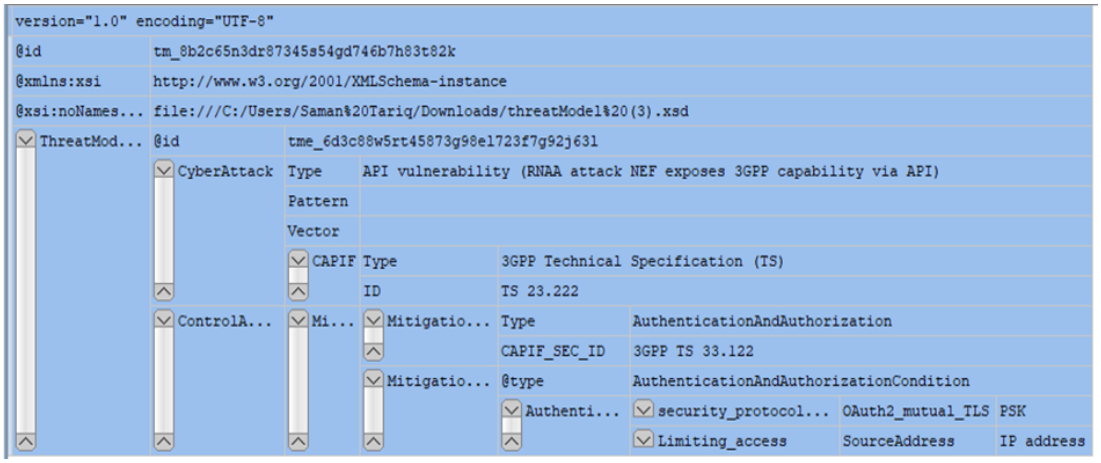


Figure 10 API Vulnerability attack XML representation

3.1.1.2 Task Force #2

The main modules involved in this Task Force are the Detection and Mitigation Engine (DEME) and Distributed Trustable AI Engine (DTE).
The DEME module is responsible for analyzing and processing network features to quickly detect a possible threat and communicate it to the DTE module.

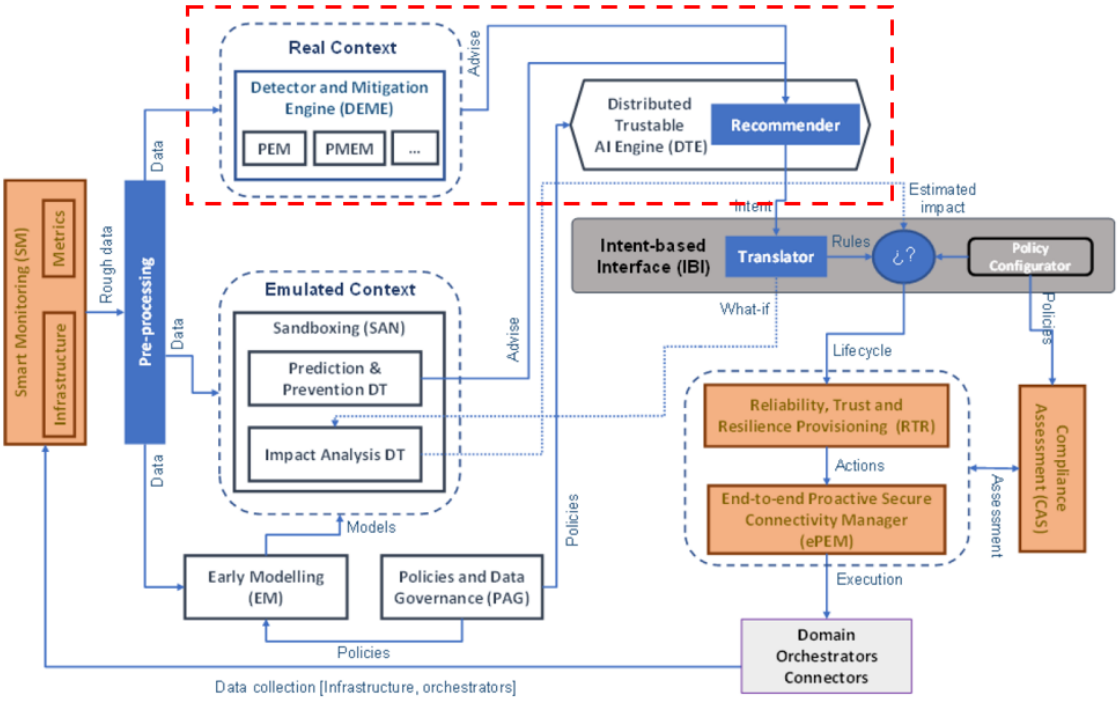


Figure 11 Modules Architecture

DEME is structured as a chain or pipeline of ML stages that can allocate different combinations of algorithms, also open to future research and testing activities (already today we are testing solutions that include innovatively customized algorithms).
Threat detector optimizes performance, including, because it is crucial, the detection time in order to have a wider range of detection even towards less standard forms of attacks, as we are unfortunately expected to face in the 6G context, such as combined attacks or new attack formulations.
For the HORSE project, for demonstration purposes, it was decided by the community to focus attention on a limited number of PoCs chosen among the most significant ones either for their dangerousness (such as the Reflection/Amplification DDoS) which by specialized literature are considered among the most dangerous even in the looming 6G context, or for their relevance in the same context (e.g. PFCP).
DEME receives from the pre-processing module, every fixed and agreed period, a data snapshot containing counters of features related to the specific attacks. As agreed, the received features for each network instance are the counter of packets NTP-monlist (NTP packets with type 7), the counter of DNS packets received every second by DNS server, and the counter of PFCP Session Deletion Requests.
Based on the above-mentioned data, on the network topology and on the historical data of the same network features collected in a non-attack state previously provided to the module, DEME is able to estimate a possible threat for each network instance. Then, DEME produces as output, for each network instance, the name of the identified attack and the accuracy of the detection. That can be retrieved by DTE module.
Once a threat is identified, a list of mitigation actions referring to the identified threat is processed.
Communication among the different modules is achieved through Application Programming Interfaces (APIs).

3.1.1.3 Task Force #3

The main modules involved in this task force are the Smart Monitoring, the Policies and Data Governance (PAG) and the Distributed Trustable AI Engine (DTE). The focus is the PAG component in action, and the Task Force uses existing datasets (not generated inside the HORSE project). The general picture of this Task Force can be seen in Figure 12, highlighted in yellow.

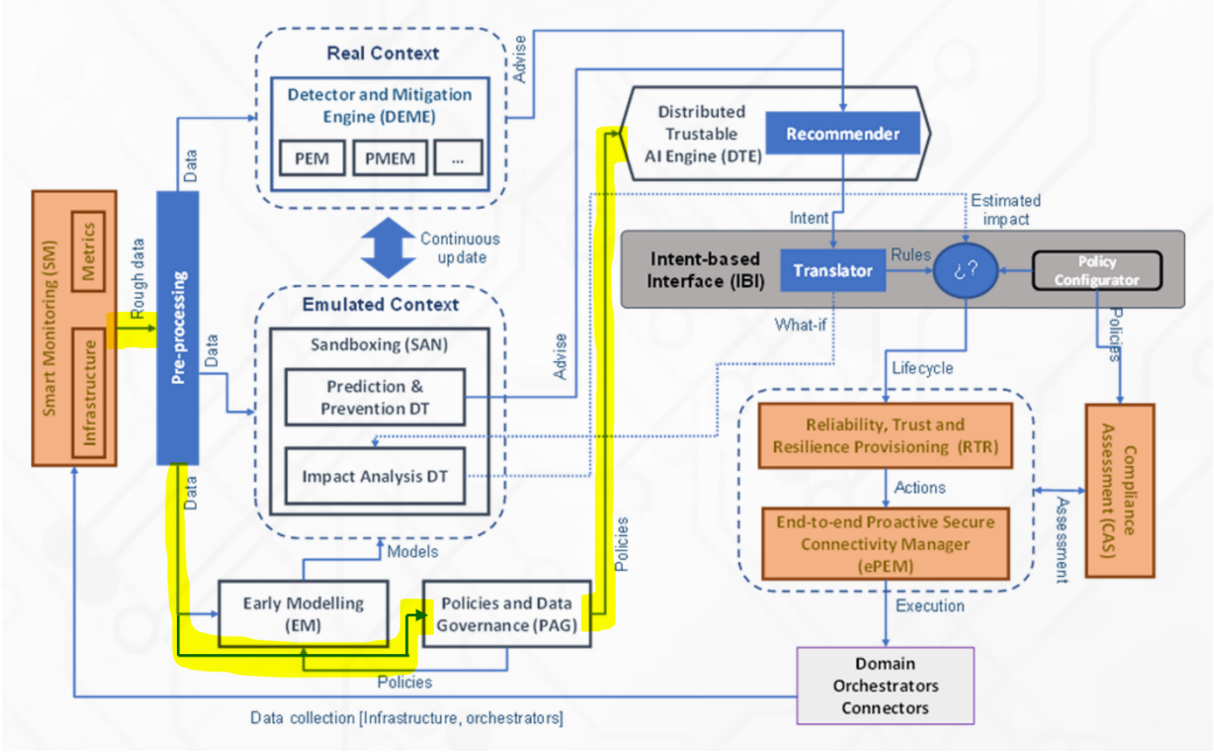


Figure 12 HORSE detailed architecture focused on the Smart Monitoring, Policies and Data Governance and the Distributed Trustable AI Engine

The initial release of the PAG component was made available on M13 of the project, as a standalone component. It implements the functionality for defining and enforcing access management rules, also defining and enforcing data retention rules, and implementing data anonymization on the collected datasets. The initial release of the PAG also includes the PAG storage, for storing the metadata and the policies.

The datasets used include: (a) sample data regarding VNFs that belong to a simulated 5G Network, and (b) 5G-NIDD, a fully labelled dataset built on a functional 5G test network that can be used by those who develop and test AI/ML solutions.

The key datasets that will be used in PAG implementation are included in the following Table 6:

Dataset name	Description	**Key features**

5G-NIDD[3]	Intrusion Detection Dataset Generated over 5G Wireless Network	Duration, runtime, tool of attack, attack type
AWID3 Dataset[4]	Wireless dataset containing multi-layer and modern attacks including Krack and Kr00k	Time, source, destination, protocol, length
TonIoT[5]	The datasets include heterogeneous data sources collected from Telemetry datasets of IoT and IIoT sensors, Operating systems datasets of Windows 7 and 10 as well as Ubuntu 14 and 18 TLS and Network traffic datasets.	Date, time, attack label, attack type
UNSW-NB15[6]	dataset created by the IXIA PerfectStorm tool in the Cyber Range Lab of UNSW Canberra for generating a hybrid of real modern normal activities and synthetic contemporary attack behaviors	Source IP address, source port number, destination IP address, destination port number, attack category

Table 6 Dataset for PAG

The datasets are stored in the Smart Monitoring module, which shares necessary information on the datasets and allows access on these datasets through custom APIs. The PAG resolves the access policies (which may be per user, or per component) and gives the details to the DTE to access the datasets themselves. Then, the DTE could, for example, identify a dataset as malicious at some point (or identify a "data store" node as malicious) and ask the PAG to block access to this dataset (any action on this dataset is then prevented).

[3] *Samarakoon, Sehan, et al. "5g-nidd: A comprehensive network intrusion detection dataset generated over 5g wireless network." arXiv preprint arXiv:2212.01298 (2022).*

[4] *https://icsdweb.aegean.gr/awid/download-dataset*

[5] *https://research.unsw.edu.au/projects/toniot-datasets*

[6] *https://research.unsw.edu.au/projects/unsw-nb15-dataset*

3.1.1.4 Task Force #4

The main modules involved in this Task Force are the Intent-Based Interface (IBI) and the Reliability, Trust, and Resilience Provisioning (RTR). Additionally, we also discussed the integration of the IBI module with the Distributed Trustable AI Engine (DTE). As defined in the previous HORSE specification of the HORSE Architectural Design [1] we favor the adoption of RESTful architectures for inter-module communication. The general picture of the architecture can be seen in

Figure 13, where the interfaces defined in Task Force 4 are highlighted in red.

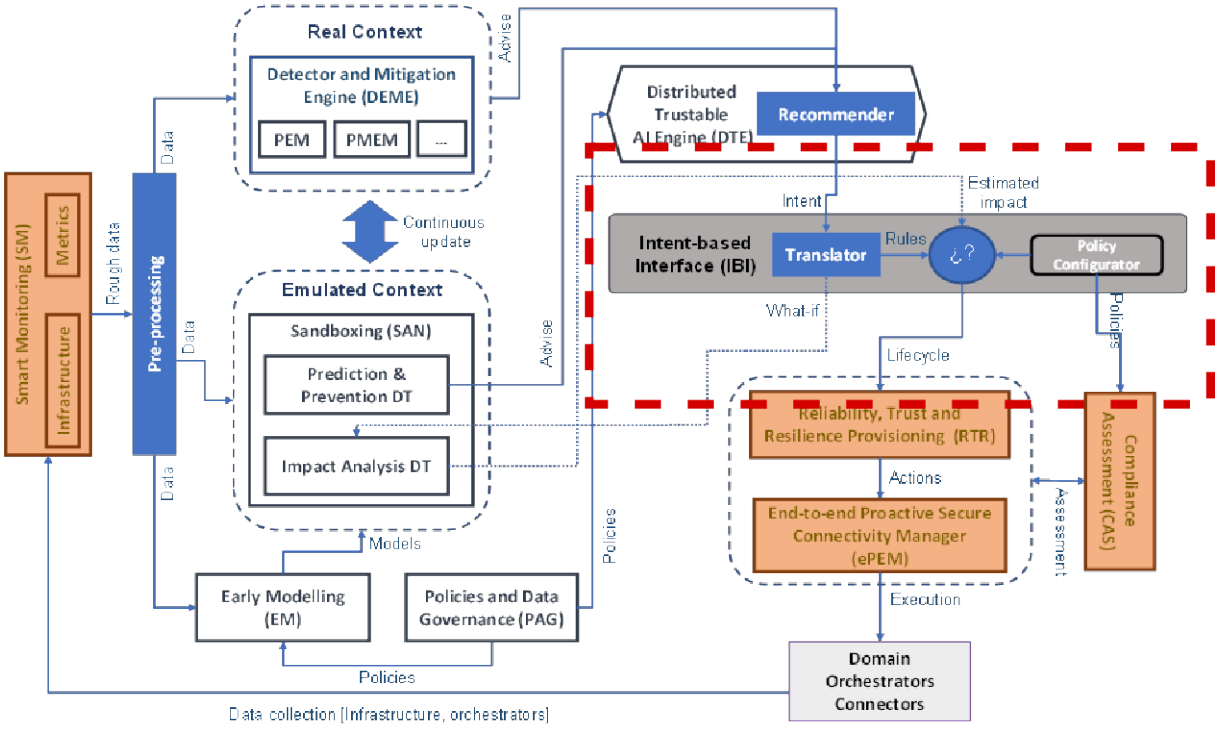


Figure 13 HORSE detailed architecture highlighting the integrations between the DTE, IBI, and RTR and its interfaces.

The IBI component receives an intent from the DTE module as input. This intent is sent by the DTE module when a threat is detected in the network. The information sent from the DTE to the IBI module helps the IBI module decide which mitigation actions should be applied in the network to mitigate the detected threat. It is worth mentioning that this information is part of the automation loop responsible for informing the IBI module of whether the selected mitigation action was successful. The data received from the DTE includes the intent and threat type, the list of functions or hosts suffering the attack identified by their IP addresses, and the duration that the intent should remain valid in the network, as depicted in Figure 14.

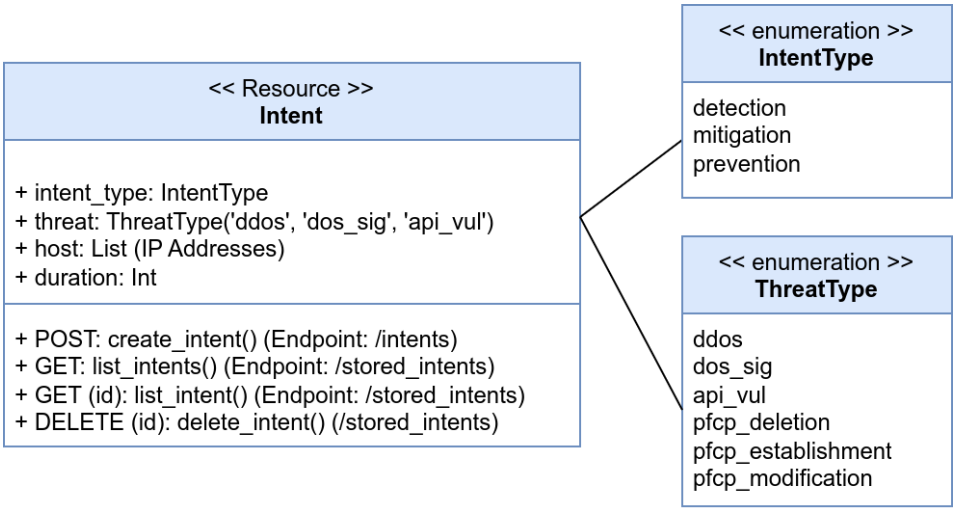


Figure 14 Messages exchanged between the DTE and IBI modules and related RESTful endpoints.

Following the workflow, the IBI module proposes a set of actions that should be applied in the network according to the information received from DTE and the intent of the network manager. The RTR module, responsible for applying prevention or mitigation actions to network elements, gets the set of actions as a collection of Workflow Units. The Workflow Unit contains the command (add or delete a Workflow Unit to the network), the intent type, the threat type, the attacked host (same as the host field in the message described above), plus three additional fields: the mitigation host (i.e., where RTR should effectively apply the mitigation or prevention actions), the action (a string encoding the action that the RTR module should enforce) and the intent identification used to identify the Workflow Units that the RTR should withdraw from the network when they are not required anymore. Figure 15 depicts the Workflow Unit message that will be sent to the RTR module and its related enumerations and types.

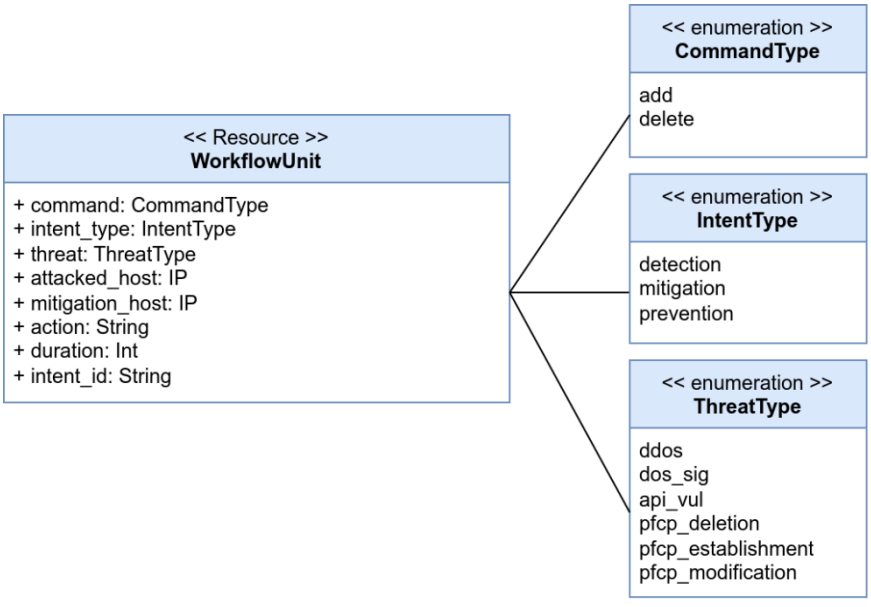


Figure 15 Messages exchanged between the IBI and RTR modules.

Finally, Figure 16 depicts the entire workflow covered in Task Force 4 as a sequence diagram, including messages exchanged between the DTE and the IBI modules and messages exchanged between the IBI and the RTR modules. The figure partially illustrates the threat detection workflow in HORSE architecture and shows the interactions of modules highlighted in

Figure 13

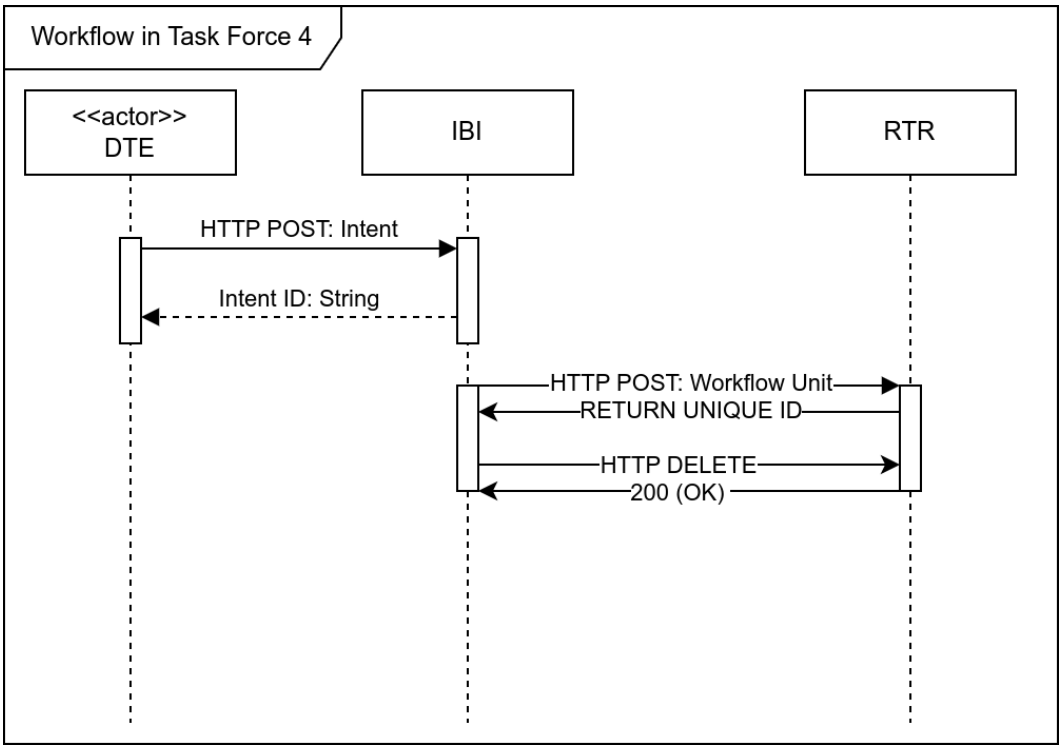


Figure 16 Sequence diagram and respective messages produced by components within the scope of Task Force 4.

3.1.1.5 Task Force #5

The main modules involved in this task force are the Domain Orchestrator Connectors (DOC) and the End-to-end Secure Connectivity Manager (ePEM), as it was explained in previous HORSE Architectural Design [1], that components enable HORSE context to enforce the mitigations actions in the different type of infrastructure.

The general picture of the architecture can be seen in Figure 17, where the interfaces defined in Task Force 5 are highlighted in red.

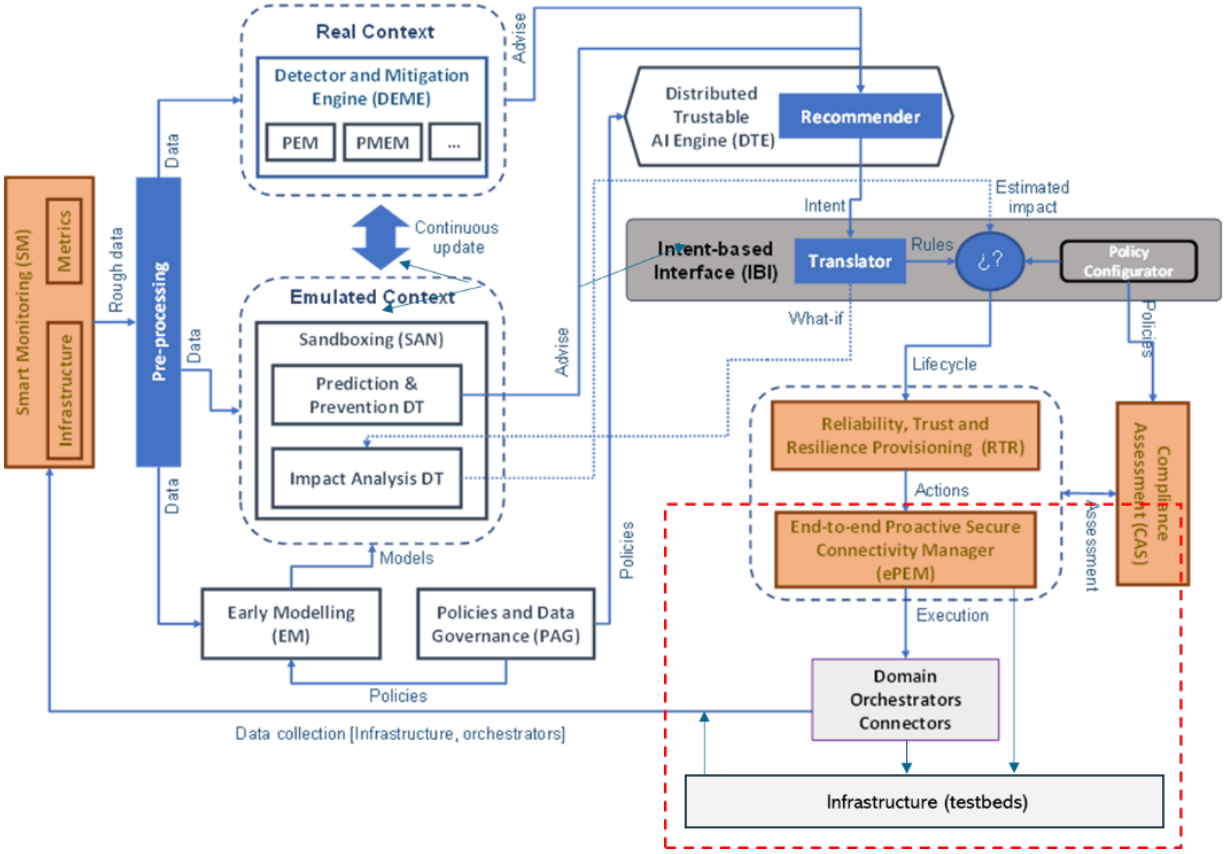


Figure 17 Modules Architecture

The ePEM decides if the mitigation actions received by RTR component should be applied by DOC or directly by ePEM. This logic determines whether the target host specified in the mitigation action, such as Public Cloud, OCM manager, Kubernetes cluster, or others, resides within the ePEM infrastructure. If the target is within ePEM, the ansible playbook received by ePEM will be executed directly on the target or targets.

In the scenario where the targets are not part of the ePEM infrastructure, ePEM undertakes a different approach. Upon receiving the mitigation action from RTR, ePEM processes the action ID, target, and action type extracted from the ansible playbook. Instead of executing the playbook directly, ePEM initiates communication with DOC through the REST API.

Through this communication channel, ePEM conveys the relevant details of the mitigation action to DOC, enabling further evaluation and decision-making. This process ensures that even when the targets are external to the ePEM environment, seamless coordination with DOC is maintained. ePEM acts as the intermediary, facilitating the transfer of information and ensuring a coordinated response to security incidents across the ecosystem.

Through this API DOC receives information to apply different actions in a successful way, there are some fields that are essentials to enforces the actions:

- ActionID, it is a unique identifier assigned by RTR that allows HORSE Context to track differents actions during their life cycle.
- Target, it is an IP of the component in which HORSE wants to modify through mitigation action in order to control an attack.
- Action Definition, it defines the action type and the service that is required to modify.

DOC enables to enforces mitigations actions in multidomain environments, in particular, DOC manages it using a multi-clusters infrastructure which allows that the topology can be allocated in different geographical places. That case is going to be demonstrated using the UMU testbed

and its different physical locations explained in Section 3.1.2.1. in this first demonstration scenario, will be used Liqo[7], which is “an open-source project that enables dynamic and seamless Kubernetes multi-cluster topologies”, to manage and enforce the actions received through ePEM API REST, other multicluster management systems are beginning to be explored and implemented in DOC component such an OCM.

Finally, the following figure shows the workflow between ePEM-DOC and the infrastructure for the case explained above.

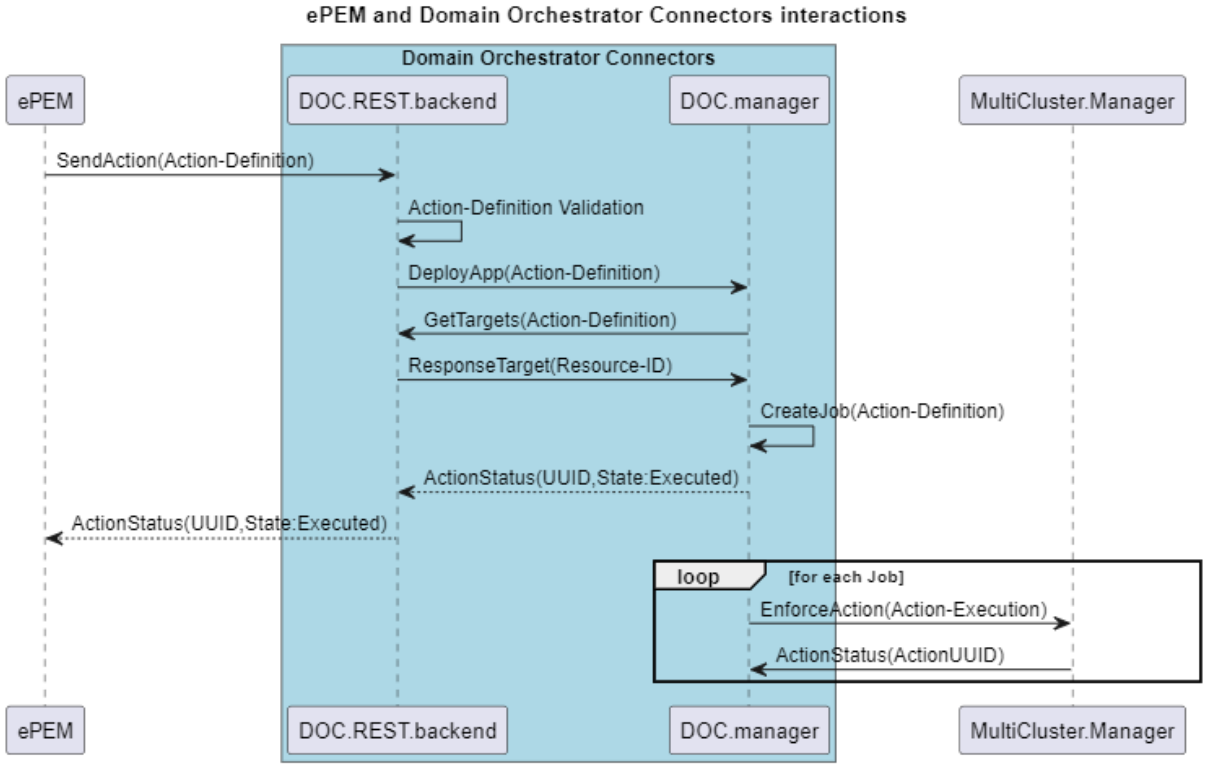


Figure 18 ePEM-DOC workflow

3.1.1.6 Task Force #6

The main modules of Task Force #6 are the Smart Monitoring (STS) and the Pre-Processing. The Pre-Processing module acts as a vital link between the Smart Monitoring (SM), tasked with gathering raw data and the following analytical procedures, i.e. DEME, Digital Twin and Early Modeling. This way Task Force #6 guarantees a centralized and standardized data monitoring and distribution environment.

[7] *https://docs.liqo.io/en/v0.10.2/*

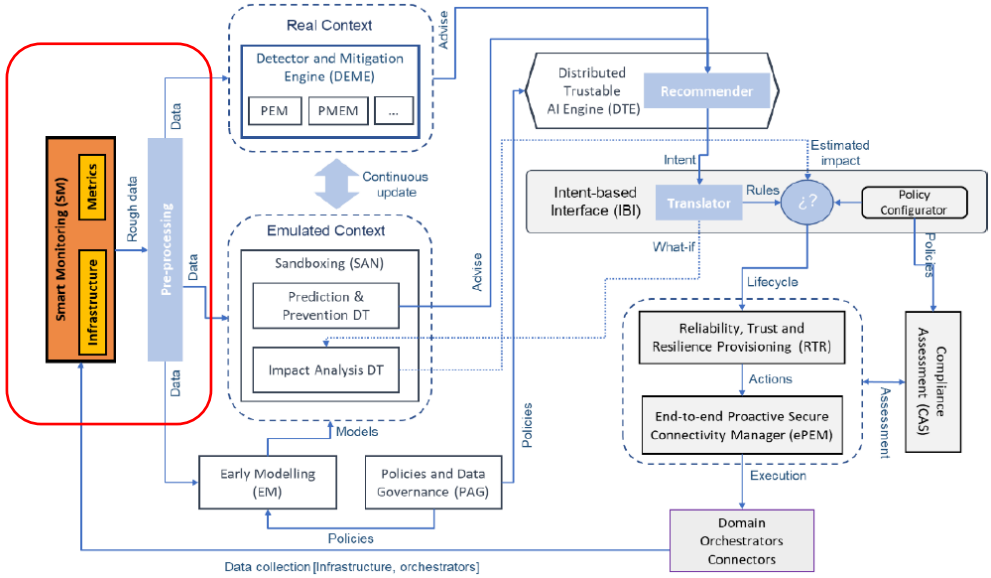


Figure 19 HORSE detailed architecture focusing on the Smart Monitoring and Pre-processing

Smart Monitoring:

To enhance data collection efficiency within the SM component of the HORSE Project, Elastic Beats will be employed as a critical tool for lifecycle management. Elastic Beats plays a vital role in collecting data for managing 6G services throughout their lifecycle. It serves as a crucial element in gathering and transmitting data to a centralized analysis platform like Elasticsearch.

Elastic Beats represents a collection of lightweight, specialized data shippers, each known as a 'Beat.' These Beats are tailored to gather specific data types from various sources. They efficiently transmit this data to Elasticsearch or Logstash for further analysis and processing. This data, once stored in Elasticsearch, becomes a powerful resource for analysis, ensuring optimal performance and security of 6G services.

Utilizing this technology will empower the SM to bolster performance, strengthen security measures, and elevate service quality within the intricate and challenging landscape of modern telecommunications infrastructure.

Smart Monitoring Security Features:

In the context of the SM, Elastic Beats usage enhances security in various ways during data collection:

- Encrypted Transmission: Elastic Beats encrypts data in transit to Elasticsearch, safe-guarding it from unauthorized access.
- Secure Authentication: It integrates with secure authentication mechanisms, ensuring only authorized Beats can transmit data, preventing unauthorized access.
- Limited Data Collection: Beats collect specific data types, reducing the risk of inad-vertently capturing sensitive information, thus minimizing potential breaches.

If we combine these aspects with the regular updates and security patches provided by the company behind Elastic Beats, along with the lightweight nature of Beats, which have minimal impact on deployed systems, we establish a reliable and secure method for collecting and storing data.

Pre-Processing:

Acting as a critical intermediary between raw data collection and the subsequent analytical processes, the Pre-processing module ensures a harmonized and standardized data

landscape. The primary data source for this module is the SM component, which collects diverse data from infrastructure components, domain orchestrators, and various metrics related to resource usage. Integration with SM's Elasticsearch facilitates real-time data consumption by the Pre-processing module. Additionally, the module integrates with the Data Fusion Framework (DFF) platform, developed by 8 Bells, to support its operations. DFF enhances functionalities related to data source management, secure data flow, and efficient data access. Integration with DFF occurs through GUI interaction and API endpoint connectivity, ensuring user-friendly data management and programmatic interaction. This approach enhances system flexibility, scalability, and automation capabilities within the HORSE architecture. Overall, the Pre-processing module's development meticulously incorporates technologies to communicate with Elasticsearch and the DFF platform, optimizing integration with Smart Monitoring data and ensuring a seamless, standardized, and scalable data preprocessing pipeline.

Pre-Processing Security Features:

The emphasis on security is especially significant within the Pre-processing module, given its pivotal role in standardizing and harmonizing data sourced from the SM's Elasticsearch database. Stringent measures are in place to protect data during collection, transmission, and storage, aligning with industry standards and best practices.

- Secure Data Transmission: Integration with the SM component ensures secure communication protocols, utilizing HTTPS encryption to safeguard data during transit.
- DFF Platform Security: DFF platform interactions, through GUI and API endpoints, enforce robust security measures. Keyrock authentication restricts access to authorized users. SSL/TLS encryption secures GUI data. Authentication methods, like token-based systems, control access to sensitive API functions, ensuring only authenticated entities interact with the Pre-processing module.
- Data Upload Validation: To ensure data integrity within the Pre-processing module, thorough validation mechanisms are utilized. Uploaded data is validated against predefined standards through schema validation. Deviations prompt alerts, preventing ingestion of potentially harmful data. Historical data auditing maintains a chronological record of uploads, enhancing traceability and accountability.
- Compliance with Data Protection Regulations: The Pre-processing module operates in compliance with stringent data protection regulations like GDPR, handling user data responsibly and promoting transparency. Data anonymization and minimization principles mitigate privacy risks and ensure compliance with evolving regulatory frameworks.

In conclusion, the emphasis on security and data collection within the Pre-processing module underscores a commitment to robust cybersecurity practices, safeguarding data integrity from collection to harmonization within the HORSE architecture. The SM and pre-processing modules collaborate to create a secure environment that minimizes data leaks as well as attempts to breach both systems. This is achieved by eliminating reliance on exposed endpoints for data flow. Such an approach mitigates potential harm or exploitation by third parties, as exposed endpoints may contain design flaws. Instead, the SM securely stores data in Elasticsearch, while the pre-processing system detects changes and efficiently forwards new data to the interested HORSE components. This seamless communication is facilitated by shared Elasticsearch credentials between the components, ensuring minimal exposure to external threats and enabling direct communication.

3.1.2 Testbed descriptions

In order to support the process of integration and validation, three testbeds are being prepared and described in the following sections.

3.1.2.1 University of Murcia (UMU) testbed

UMU will contribute to this project with Gaia5G. Gaia 5G is a laboratory and testbed, illustrated in Figure 20, for new technologies in the fields of telecommunications, network and cluster orchestration, IoT, smart cities, and security. It is part of the department of Information and Communications engineering of the Computer Faculty of the University of Murcia, physically located in the Espinardo campus. This testbed enabled by the latest computing and networking technologies, has different virtualization clusters encompassing different technologies (KVM, K8s, Openstack and hyperconvergence), has SDN enabled switches, a NFV enabler orchestrator and have different radio access technologies, from its own multiple campus-wide 5G networks, to different IoT and ITS technologies, such as LoRaWAN, 802.11p/G5 and C-V2x (PC5/Sidelink) enabled zones. All this infrastructure is self-managed and not dependent of third parties or the university, which enables the deployment of disruptive tests and use-cases.

The testbed is dedicated mainly for research purposes and is focused on the area of 5G technologies as well as virtualization, with a powerful reconfigurable infrastructure that enables researchers to investigate and demonstrate very different use cases and technologies. The physical location of the testbed in the campus is a unique opportunity to leverage the whole campus as a smart city demonstrator, a “smart campus”, as the campus is a large campus with a range of facilities, services and buildings that can be (and some are) sensorized with a plethora of IoT devices. In the campus there are smart parking spaces, smart energy management systems, occupation sensors, ambience sensors and many more that are integrated into this smart campus platform that Gaia can leverage to its benefit.

Overall, the integration of the testbed with the smart campus infrastructure provides a valuable opportunity to develop and test smart city applications and services, and to advance the development of these technologies to improve the efficiency, sustainability and quality of life.

Also, having a plethora of buildings physically next to each other but in an area of a few square kilometers to deploy infrastructure in, enabled us to have our testbed physically differentiated. Since most of the computing power and network infrastructure is located in the ATICA (UMU’s IT services) CPD, but also have a second powerful site in the Computer Faculty (Gaia laboratory room and T3 IoT laboratory). These are the two main sites for our testbed, in which we have several computing clusters, our network backbone, which is SDN and P4 enabled, and a 5G IoT laboratory.

UMU is also equipped with infrastructure in other buildings, where some edge servers have been deployed with different architectures, which allows to test true multi-domain and multi-operator use cases.

Figure 20 Gaia 5G

Main network description and virtualization capabilities

The testbed is equipped with four different virtualization platforms that enables us different approaches and tools to demonstrate and trial different technologies.

The main virtualization “stable” cluster, that mostly hosts our own internal services, such as GIT repository, network documentation and management and other services related to ongoing projects and researchers’ services is a four node Proxmox Virtualization Engine, based on Linux. It uses native technologies such as KVM and LXC for virtualization. This cluster has a total of 164 cores and 2 TB of RAM. This cluster is being enabled with GPU computing support using NVidia A100 GPU cards and can also host compute VMs for others cluster in our network. This computing capabilities are completed with 35TB of total usable RAID6 storage in total among all nodes and two NAS with 25TB of redundant storage each.

This scenario also has two differentiated (stable and unstable) Openstack rocky and yoga clusters with two nodes each and having a total of 80 vCPU and 512GB of RAM each.

Some new additions for the clusters in our testbeds are two different K8s stacks (also as stable and unstable) with 60 cores and 256GB of RAM, and two hyperconvergence clusters based on Nutanix HCX solution with 48 cores, 512GB RAM and 16TB of storage each cluster.

These main computer clusters are complimented by embedded and industrial computing nodes, some based on ARM boards (such as Raspberry Pi 4 and Mochabin SBCs) and some are AMD64 embedded industrial server that are deployed where needed, either independently or as a compute node integrated in K8s or Openstack clusters.

All these clusters are supported by a main SDN-enabled backbone based on two PicOS switches that interconnect our main sites and augmented with several P4 reconfigurable switches based on the Tofino architecture. A more detailed list on our switches follows:

- OpenFlow PicOS based 96x10G/12x40G Switching backbone
- P4 capable 16x1/10/25G, 32x 10/25G, 4x 100G
- Openflow enabled Switches with 48x10G + 6x40G
- APS Networks BF2556x1t P4 Tofino-based switch 48x25Gb + 8x100Gb (PTP capable)
- Wedge 100BF-32X 100GbE P4 Tofino-based switch 32x100Gb
- Wedge 100BF-65X 100GbE switch 65x100Gb

These diverse solutions and SDN/P4 enabled network interconnection servers as a base for NFV paradigm demonstration, and as being this paradigm one of our research interests, parts of the network and clusters are being orchestrated using OSM (Open Source MANO) and this enablement has been the basis for some of our research, such as a ZSM (zero touch security manager) orchestrator that has been a ETSI reference PoC platform.

The whole switching fabric is monitored for performance metrics using Prometheus and can be visualized using Grafana or Apache Superset and all logs are parsed and storage using an ELK (Elasticsearch – Logstash – Kibana) stack. This gives us full visibility over the infrastructure and aids in the experimentation with the testbed.

5G

Gaia 5G is enabled by 5G equipment. We have three different gNBs, two based on SDRs on our T3 laboratory and a third one using exterior 20W remote radio head using the n78 (3.5GHz) band, connected via CPRI fibre to our gNB, housed on the main CPD in ATICA. This gives coverage to most of the campus area while leaving us complementary equipment to experiment breaking changes in our laboratory. All this is backed by Amarisoft's software solutions that implements the gNB and a basic 5G SA core, but we also have deployed successfully open source 5G SA cores, such as Free5GC and Open5GS, even enabling our network orchestrator to reconfigure all the 5G network, from gNB to core, being able to deploy every element dynamically.

Over this 5G environment, two other networks were deployed on the campus for R&D purposes on the scope of a national project. Each one with its own computing and networking solutions and providing its own cloud solutions, allowing this way, to support multi-operator scenarios on the campus.

It is also available SDR and laboratory equipment to test and validate open-source RAN implementations, and a 64 UE emulator and an assortment of 5G capable modems and phones.

- 2x USRP B210 SDR - Dual Channel Transceiver (70 MHz - 6GHz) - Ettus Research
- 1x USRP N310 SDR (ZYNQ-7100, 4 CHANNELS, 10 MHZ - 6 GHZ, 10 GIGE) - Ettus Research
- 2X RRH band 78
- 3x Amarisoft licenses gNB + core
- 1x Amarisoft UE Simbox 64 (up to 64 5G UEs)
- 2x Amarisoft Callbox (AIO solution with gNB, core and SDRs)
- Assorted antennas, filters and amplifiers for SDR scenarios
- 20 assorted 5G SA capable modems (Quectel and Fibocom modules)
- Keysight NEMO Handy L2-7 5G engineering phone
- Anritsu FieldMaster MS2090 Spectrum analyzer with 5G PHY analysis capabilities

Figure 21 illustrates the overall Gaia 5G environment available for the HORSE validation.

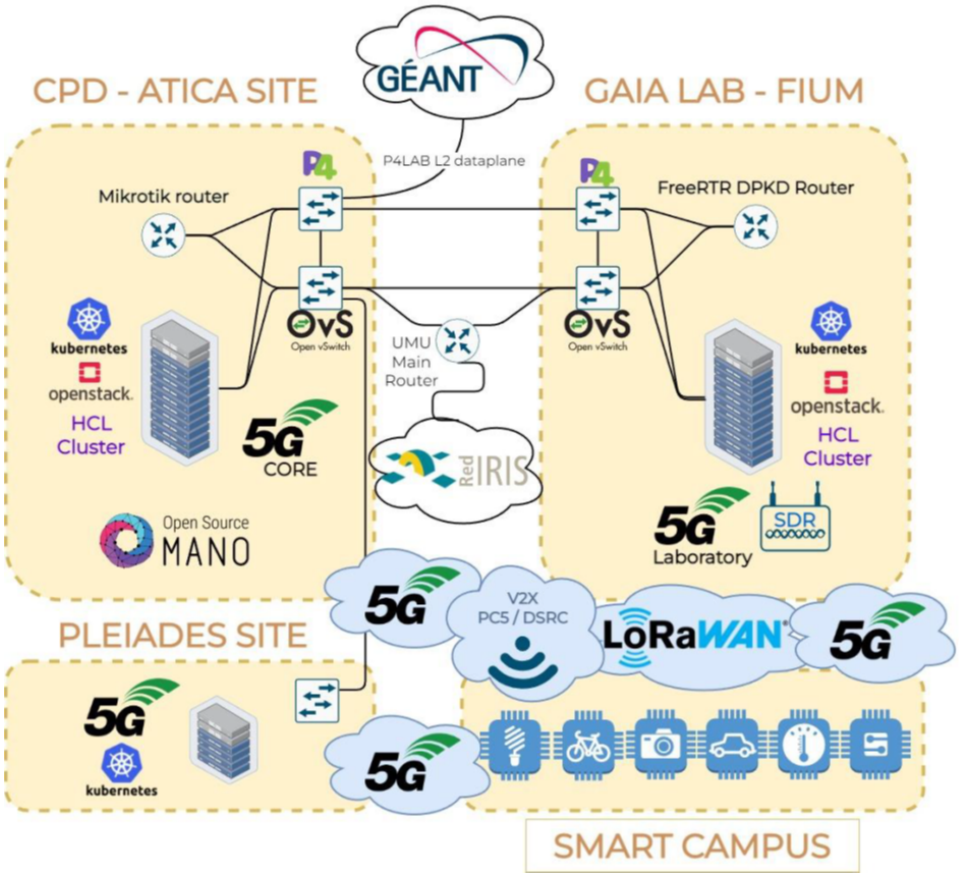


Figure 21 Gaia 5G connectivity schema

3.1.2.2 Nacional Inter-University Consortium for Telecommunications (CNIT) S2N (Smart and Secure Networks) Testbed

CNIT S2N testbed located in Genoa (Italy) is a multi-layered hardware and software facility for the advanced experimentation and demonstration of 5/6G, Edge and Cloud Computing technologies. It is specifically conceived to host multiple isolated tenant spaces, or "islands" (e.g., project environments) that can emulate complete 5/6G network environments, as well as to manage and configure their respective physical/virtual resources through a Metal-as-a-Service (MaaS) approach and the software elements through Red Hat Ansible. The testbed is part of the Scientific Large-Scale Infrastructure for Computing/Communication Experimental Studies (SLICES) [9] project, which has been selected to be part of the 2021 roadmap of the European Strategy Forum on Research Infrastructures (ESFRI) [10].

In a nutshell, the testbed has hardware capabilities for setting up 5/6G network environments, complete with User Equipment (UE), UE emulator, radio/wired access, programmable connectivity, security system and (distributed) computing domains, as well as equipment useful for performance evaluations such as traffic generators and power monitors. Based on these infrastructure resources, the testbed platform integrates both proprietary and open-source software to realize and dynamically manage the islands on top – specifically, the involved base 5/6G and application-/slice-specific network services (NSs), (distributed) computing domains and WAN overlay networks.

The testbed infrastructure, shown in Figure 23, is an integration of both general- and special-purpose equipment to realize underlying 5/6G networks substrates and relative slices, as well as edge-cloud computing resources.

- At the infrastructure level, the testbed currently hosts the following equipment:35 servers (1300 cores, 10 TB RAM, local & central SSDs/SAS storage >100TB),
- 8 high-speed switches (918 ports from 1GbE to 100GbE),
- Base stations: 2x Amarisoft Callbox 5G gNB MIMO 4x4, + 3x LTE+ eNBs (USRP-based),1x O-RAN sub-6GHz RU, 1x O-RAN mmWave RU, 2x 1x O-RAN sub-6GHz AiO femtocells
- 1x Amarisoft 5G UE Emulator
- 4x GPU Nvidia A100
- 1x P4 (Tofino-enabled) switch
- Power monitors, hardware traffic generators, hardware firewalls, 12 UEs (drones, tablets, modems, etc.).
- The test bed is flexibly managed through a Metal-as-a-Service approach (OpenStack and Kubernetes instances as-a-Service)
- Site-to-site and client-server VPNs.
- Testbed-wide time synchronization through IEEE-1588v2 PTP Grandmaster Clock (approx. 20 ns precision)

Thanks to these assets, the testbed can currently deliver a complete 5G SA solution and research activities are in place to evolve beyond 5G. Furthermore, the hardware assets are complemented with a set of software products that allow for the deployment and management of vertical applications and related slices according to IaaS and PaaS paradigms and to even create and handle wide-area networks: in fact, not only different VIMs and Kubernetes clusters can be created, but it is even possible to view the available computing resources as edge or cloud thanks to the presence of a delay generator. The main software artifacts are the following:

- OpenStack, Open-Source MANO (OSM), Kubernetes, etc.
- CNIT S2N/Unige TNTLab OSS
- 4G/5G and networking VNFs
- DPDK-based delay and packet loss emulation
- ONOS SDN Controller (P4 and OpenFlow)
- Vertical Application Orchestrator
- Wide Area Infrastructure Manager
- Observability Stack (Prometheus, ELK, etc.)

Figure 22: The CNIT S2N testbed.

- Keysight IxCore
- Automated and personal VPNs.

3.1.2.2.1 The Physical Infrastructure

Figure 23 shows the hardware resources available in the testbed.

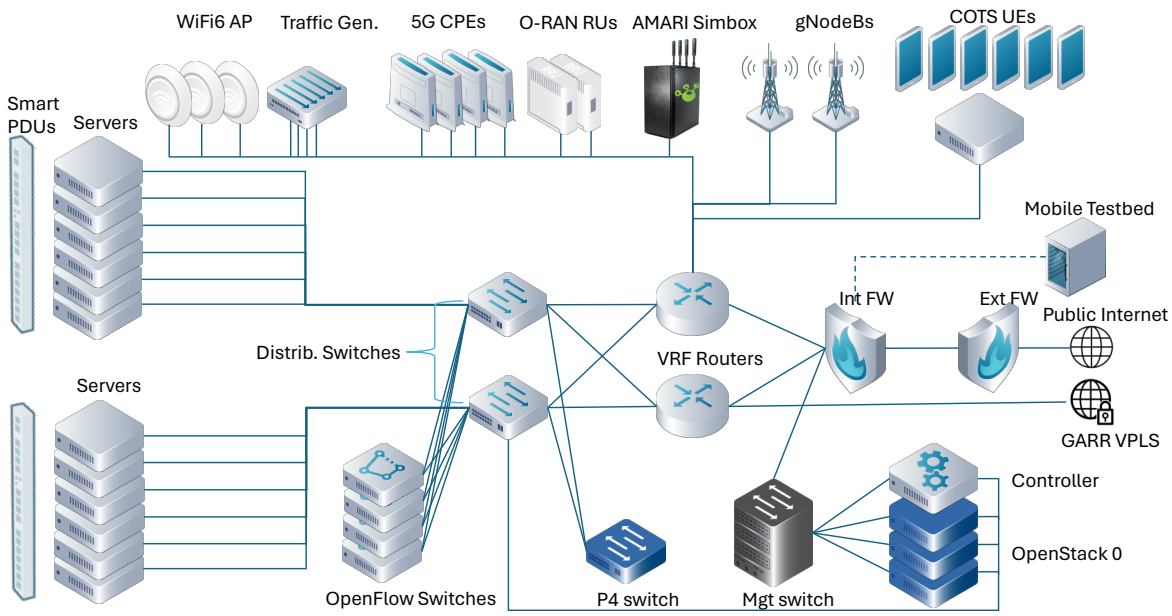


Figure 23: The physical infrastructure of the S2N testbed.

The stack of servers on the bottom right represents the central nervous system of the infrastructure and is the only persistent installation of the testbed. In fact, the other resources, both physical and logical, are delivered to the experimenters on-demand, in an as-a-Service fashion. The controller performs resources' assignment and network configurations, while the OpenStack 0 VIM (OS0 hereinafter) offers several services that are common to all experiments and are delivered by the management switch (dark gray left to the Controller in Figure 2) through an out of band management network. OS0 also hosts the demilitarized zone (DMZ). The actual operations for creating the isolated islands for the experimenters are described in the next section.

On the left of Figure 23 are depicted two stacks of servers that correspond to the racks in Figure 22 and represent the programmable computing resources of the testbed. Along with the computing resources, several OpenFlow switches and one P4 switch [11] are made available for the experimenters, as well as access and user devices. Access devices include IEEE 802.11v6 access points, 5G CPE routers [12], O-RAN radio units, as well as two physical gNodeBs. Regarding UEs, we have an Amarisoft UE simulator [13] and several 5G smartphones. The latter are made available remotely thanks to a NUC that hosts Guacamole [14] and screencpy [15] to mirror the screens via HTML. The set of resources made available for the experimenters is completed by a traffic generator and two PDUs.

The physical topology of the testbed is realized through a network of switches with speeds spanning from 10 to 100 Gbps. The connectivity of the islands is handled through VRFs dedicated to each island, whose isolation is made possible by the internal firewall: all traffic from an island is forwarded to this firewall that prevents its exchange with the other islands via specific rules. The external firewall instead is used to manage the connectivity towards the

public Internet. In addition, a direct link to the national research network (GARR) [16] will be established shortly to interconnect the testbed with the other facilities involved with the SLICES Infrastructure.

The entire physical infrastructure is monitored by means of an observability framework that exploits Simple Network Management Protocol (SNMP) interfaces and Prometheus exporters, collects the results into time series and exposes them to the experimenters through a database made available on the OS0 DMZ.

3.1.2.2.2 Management of the Infrastructure

CNIT S2N testbed offers customizable experiment spaces called "islands" and a meta-orchestrator is also available. A special system creates these "islands" on-demand. It allocates servers, sets them up, and manages their lifecycle. This lets researchers build unique network environments for secure and tailored experimentation. The meta-orchestration solution is composed of two main components, namely the Metal Convergence Layer (MetalCL) and the NFV Convergence Layer (NFVCL) [17] .The testbed utilizes Virtual Routing and Forwarding (VRFs) to assign devices (phones, switches, etc.) to isolated experiment environments called "islands." Each device connects to a VLAN within its island's VRF, allowing it to share routing information and be accessible by researchers. MetalCL, the core management system, goes beyond our testbed. It can handle multiple geographically separated facilities as "zones," each with unique hardware, software, and functionalities. MetalCL adapts to each zone's capabilities, offering various levels of control over resources. Imagine a toolbox with different tools for each zone (Figure 24).

- Limited Control (No programmable domains): Users can choose from pre-configured network slices and processes with no customization options.
- Some Configurability (Configurable Domains): The system allows limited configuration changes after a feasibility check.
- Cloud-Native Management (Domains programmable at PaaS): Users manage cloud-based services within a pre-built Kubernetes cluster.
- Infrastructure Management (Domains programmable at IaaS): Users can build and manage both cloud infrastructure (IaaS) and platform services (PaaS) for networks.
- Full Customization (Domains programmable at hypervisor OS): This offers the most control, allowing users to install and customize the entire cloud infrastructure.
- Ultimate Control (Domains programmable at MaaS): This advanced option, available in our testbed, grants complete control over physical resources like servers and network devices for dynamic infrastructure creation and management.
-

In essence, VRFs provide network isolation for experiments, while MetalCL offers various control levels over resources within each zone, from pre-configured options to full infrastructure customization.

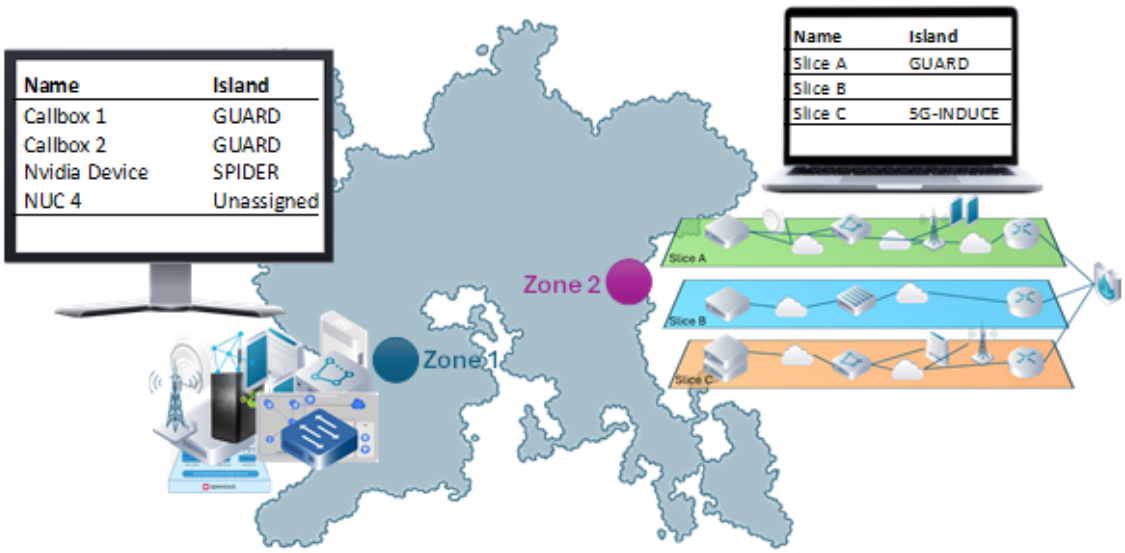


Figure 24: Example of two zones over a geographical area with different programmability levels, Zone 1 exposing programmability at MaaS level and Zone 2 exposing a catalogue of slices.

After MetalCL sets up an island, a project is created for its specific management software. This project is separate from shared software used across all islands. Notably, the NFVCL manages the entire lifecycle of network services within the island. It can build and orchestrate complex network environments (like a 5G network) by combining different virtual and physical network functions. The NFVCL uses "blueprints" to define these network environments and can handle multiple blueprints and instances for different needs. These blueprints dynamically choose the specific network functions needed based on user requests.

The NFVCL manages the lifecycle of network services within an island. It first defines resource needs and tells a higher-level orchestrator where to place network functions. Then, it creates configuration files and applies them to the network functions. The NFVCL can even manage Kubernetes clusters for container-based network functions. Figure 25 shows the entire process.

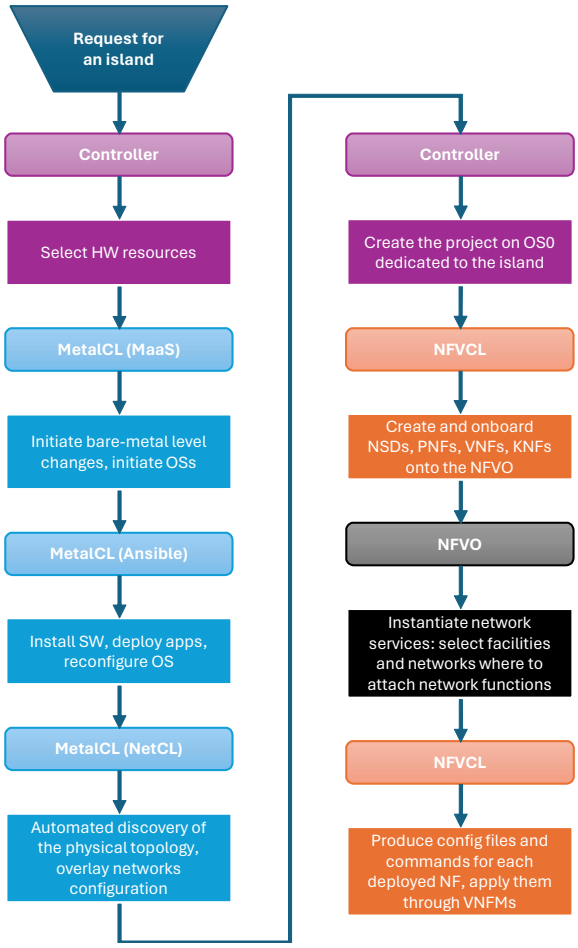


Figure 25. Flow diagram representing the steps performed by the MetalCL and the NFVCL to setup the infrastructure component and networking of an island in the S2N testbed.

An example island (Figure 26) has three networks: a "Control Network" for management, a "Mgmt Network" for internal communication between network functions, and a "WAN" used by OpenStack within the island. Within an island's OpenStack project (Figure 27), a 5G core network can be deployed with a physical or emulated base station. Network functions can be shared or customized, even creating new ones, to meet the experimenter's specific needs.

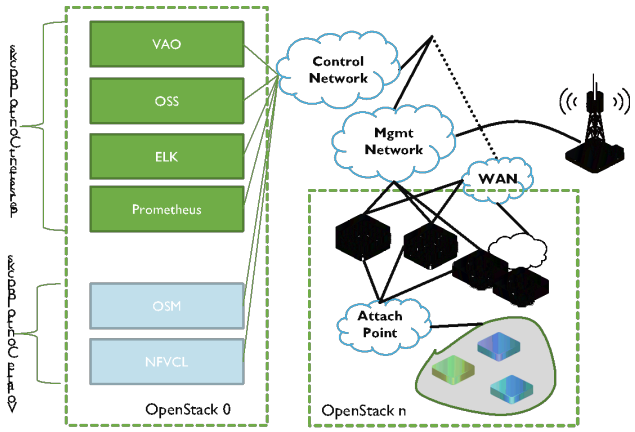


Figure 26: Example of an island architecture and interfaces.

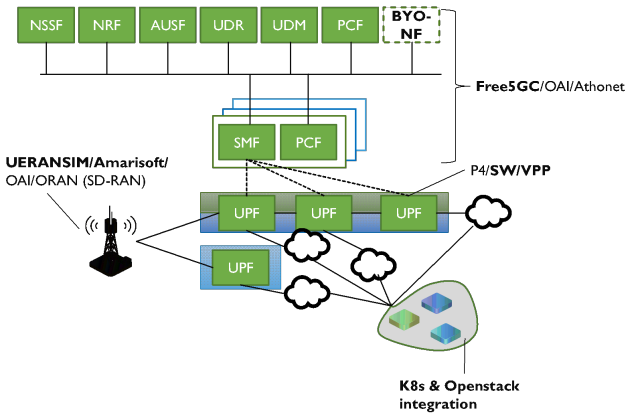


Figure 27: Deployment of a 5G core in the example island

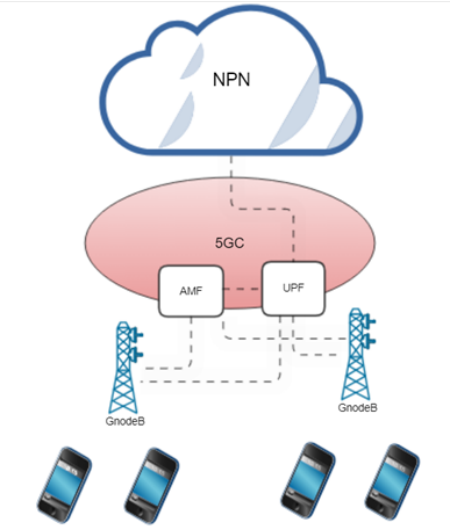
3.1.2.3 UPC (University Polytechnic of Catalunya) testbed

The CRAAX's testbed is a recently ongoing effort aimed at setting a hardware and software facility for the advanced experimentation and demonstration of 5G and B5G, Edge and Cloud Computing technologies. It is part of the Computer Architecture department of the University Polytechnic of Catalunya, physically located at the CRAAX Lab in the EPSEVG (Polytechnic School of Engineering of Vilanova I la Geltrú) campus. The testbed has hardware capabilities for setting up 5G network environments, complete with User Equipment (UE) and antennas. This section describes the 5G, edge-to-cloud architecture logically and physically, describing the interconnection between the different components.

Logical architecture

The logical architecture for the testbed is illustrated in Figure 28, which presents a diagram with the main components of the architecture:

- **NPN (non-public network) / Cloud**: It will oversee hosting the applications for 5G.
- **5GC**: Is the central part of the 5G network architecture. It is responsible for managing the network functions and services.
    - *AMF(Access and Mobility Management Function)*: manages the mobility of the user equipment (UE) and provides access to the 5G network.
    - *UPF (User Plane Function)* : handles the user data traffic.
- **UE**: User Equipment is a device that is used by the end-user to access the 5G network.



- **NPN (non-public network) / Cloud**: It will oversee hosting the applications for 5G.
- **5GC**: Is the central part of the 5G network architecture. It is responsible for managing the network functions and services.
    - *AMF(Access and Mobility Management Function)*: manages the mobility of the user equipment (UE) and provides access to the 5G network.
    - *UPF (User Plane Function)* : handles the user data traffic.
- **UE**: User Equipment is a device that is used by the end-user to access the 5G network.

Figure 28 5G tested – logical architecture

Physical architecture:

The physical description of the testbed architecture describes the different hardware components used in the architecture, how they are interconnected, and the different networks defined.

- **Network1:** This network is composed of UE devices (Smartphone and SIM), connected to the gnodeB antenna of the amari callbox classic 1.
- **Network2:** This network is composed of UE devices (Smartphone and SIM), connected to the gnodeB antenna of the amari callbox classic 2.

- **Network3:** In this network resides the 5G core (callbox classic 2). Both gnodeB antennas are connected to the 5GC and the 5GC is connected to the router. A server with Open5GCore also will reside in this network.
- **Network4:** In this network, a Kubernetes cluster has been implemented, where different applications for the 5G network can be deployed.
- **Network5:** This network hosts the SDN controller, to be able to deploy the functionalities of a Software Defined Networking.
- **Network6**: In this network can be emulated a gnodeB and a UE.

All these networks will be interconnected by a router, switch and the 5G core as depicted in in Figure 29.

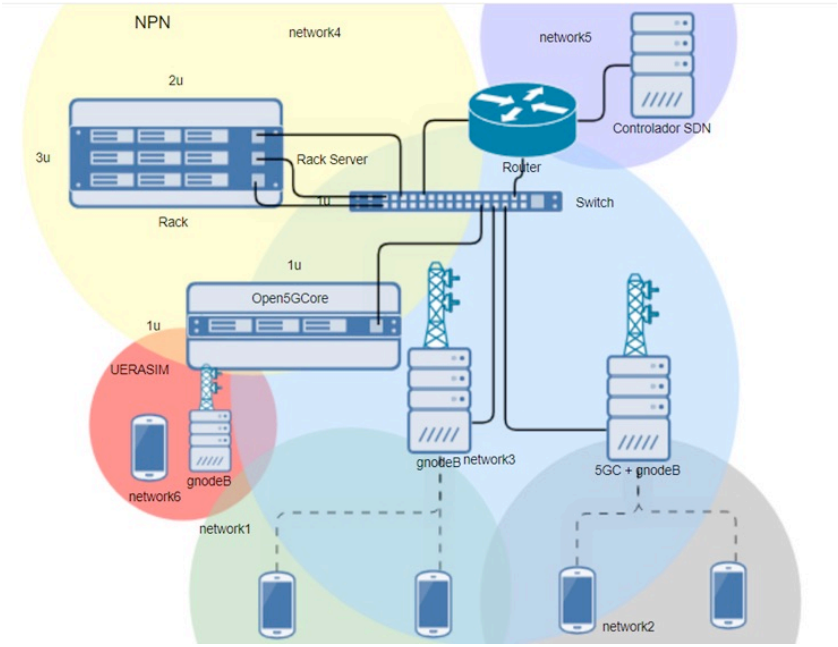


Figure 29 5G testbed – physical architecture

The hardware and software used to deploy the 5G testbed is described below.

Software and Network

On the 4G/5G networks domain:

- Real network with 5G core and gnodeB provided by two classic Amarisoft callboxes. In addition, it provides 5G core emulation with open5Gcore.
- For UE, 10 SIMs are provided by Amarisoft, with the possibility to emulate devices and antennas with UERASIM.

On the cloud domain:

- A Kubernetes cluster with real servers with "high servers" characteristics, consisting of a master and the possibility of having between 2 and 4 worker nodes “high”. In addition, "low servers" workers can be added.
- A server for virtualization with Proxmox and the possibility of extending it to a cluster.

On the network domain:

- 2 "Low servers”. One of them is the main router that handles all communications. The other one is a SDN server in charge of managing all the networks.
- A Cisco switch for the different connections.

Hardware

In the testbed, different types of servers can be used according to specific requirements. Table 7 summarizes them:

	High servers	Medium servers	Low servers
CPU	2 CPU Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz	2 CPU: Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz	1 CPU: Intel(R) Xeon(R) CPU E5504 @ 2.00GHz
RAM	125 Gb	94Gb	8Gb
SSD	2Tb	2Tb	150 Gb

Table 7 Servers' specification

In addition, the testbed has a Cisco-Catalyst 2960-X Series switch and 2 Amarisoft Classic callboxes with the specifications summarized in Table 8.

Features	Specification
4G LTE	3 cells 20MHz 2x2 1 cell 20 MHz 4x4 + 1 cell 20MHz 2x2
5G NR SA Mode	1 5G cell 50MHz 2x2 3 cells 20MHz 2x2 3 cells 40 MHz SISO
5G NR NSA Mode	1 5G NR 50MHz 2x2 + 1 LTE 10MHz 2x2 1 cell 5G NR 40 MHz 2x2 + 1 cell LTE 20 MHz 2x2 1 cell 5G NR 40 MHz SISO + 2 cells LTE 20 MHz 2x2

NB-IOT	3 NB-IoT standalone cells
LTE-M	3 LTE cells with an in-band or guard-band NB-IoT cell each 3 LTE cells with CAT M1 support

Table 8 Amarisoft Classic callboxes specification

The testbed will incorporate ONOS or open daylight for SDN, and Katana for "Network slicing is a 5G cutting edge technology that enables the creation of multiple virtual networks on top of the shared physical infrastructure.

3.2 Verification & Validation (V&V) methodology and tooling

The V&V will be progressively updated and refined according to the deployment of the HORSE releases and the available features. For the first release, a complete mapping (see section 2.1) of the HORSE features against the available testbeds was performed and the demonstrations scenarios (Table 9) were well-defined for validation. For the first release, the V&V will be oriented to functional tests involving the HORSE modules while future releases will be also oriented to performance.

3.2.1 KPIs and procedures for their measurements

After the first IT-1 will be released the validation phase will be initiated (Figure 34). This validation phase is oriented to assure the HORSE functionalities, and the validation procedures are based on demonstrations (see section 3.3.2 where the demonstrations are described). This is important to mention that a task on the scope of WP5 is still ongoing to identify and characterize the KPIs for assessing the applicability and the performance of the HORSE platform. Therefore, related to future releases, a complete list of KPIs and a list of tests will be defined as well as the procedures that will lead to collect measurements.

3.2.2 Tooling related to CNIT S2N, UMU and UPC testbeds

This section is focused on the identification and also in the description of the main tools that are available in the testbeds and can be useful to support de setup and the validation phases.

3.2.2.1 Tools in the CNIT S2N testbed

3.2.2.1.1 Service Monitoring

Service monitoring has always been crucial for telecom operators to troubleshoot networks and ensure compliance with customer service level agreements (SLAs). As telecom transitions

to virtualized environments, real-time observability becomes even more critical as a key requirement for DevOps operations. In the context of 5G applications being implemented through microservices, the associated VNF components (VNFCs), packaged as containers, can be frequently initiated, stopped, updated, scaled, and migrated.

A monitoring and observability system must possess the capability to adjust to alterations automatically and consistently in the deployment and communication of microservices. Consequently, current monitoring frameworks will need to undergo evolution to meet the following criteria: adaptability, responsiveness, scalability, customization, manageability, distribution, isolation of multiple tenants, punctuality.

Several observability systems were created for monitoring virtualized environments. For example, Docker provides an Application Programming Interface (API) for real-time resource utilization metrics. However, these solutions fall short in meeting all the necessary requirements for monitoring VNF/CNF instances in 5G systems.

3.2.2.1.2 Traditional Monitoring

Contemporary Information Technology (IT) systems and following mobile networks have grown in complexity in four specific ways: modular (composition of different services and layers), distributed (computational and geographical), dynamic (MEC) and ephemeral (development pace in mobile networks). The rising complexity in this composite structure presents a significant challenge for traditional monitoring technology. These four dimensions of complexity indicate that IT systems are transforming to include a growing variety of diverse and independent components. In the past, monitoring technologies were tailored to the characteristics of the components (e.g., networks, servers, storage and applications) being implemented separately and monitored [18].

The decision to specialize was influenced by two key factors. First, different components are generating distinct data types in specific scenarios, making it logical to customize data ingestion technologies based on the context of data collection. Second, these components are interacting infrequently and unpredictably, making it unnecessary to integrate observations from one component with those from another. This meant that troubleshooting for root cause analysis rarely required looking beyond the immediate context.

Yet, the increasing complexity of IT infrastructure has diminished the original reason for specialization. The multitude of diverse components has exceeded the ability to monitor them individually. Architectural layers have multiplied, and the role and behavior of components within each layer, such as containers or non-containers, have undergone significant changes. Importantly, the principles for interpreting the self-describing data generated by one component cannot be universally applied to another monitoring in static and monolithic applications presents a relatively straightforward challenge. However, in microservices, the complexity of connections among diverse components has significantly increased due to multiplication and differentiation. The interdependence between components has surprisingly strengthened. Consequently, the cause of a performance issue in one component often originates from a state change in other components.

A monitoring approach tailored to a specific component would offer only local information, which might not be adequate for an operations team to identify the cause of a problem, let alone predict its recurrence in the future [19]

3.2.2.1.3 Observability

As distributed IT systems and application patterns rapidly evolve, the operations management domain has witnessed the emergence of various new terminologies and approaches. A particularly crucial concept among these is observability which is defined as “the ability to

measure the internal state of a system only by its external outputs" [20]. These external outputs are primarily known as telemetry data such as logs, traces and metrics.

Observability offers both high-level overviews of the system's health and detailed insights into implicit failure modes. In DevOps, monitoring and observability are often discussed together; while both aim to ensure system reliability, they possess a subtle distinction and are interlinked. Fundamentally, observability does not substitute monitoring, nor does it make monitoring obsolete; rather, they complement each other. Figure 30 depicts the differences between observability and monitoring systems.

Monitoring focuses on tracking a system's health through a set of predefined metrics and logs, aiming to detect specific types of failures. However, in distributed systems, changes are dynamic and often occur simultaneously and regularly. This can lead to issues that do not fit the predefined categories, causing them to be missed by the monitoring system. Observability, on the other hand, builds upon monitoring and helps in understanding where and why a failure occurred, along with what triggered the issue.

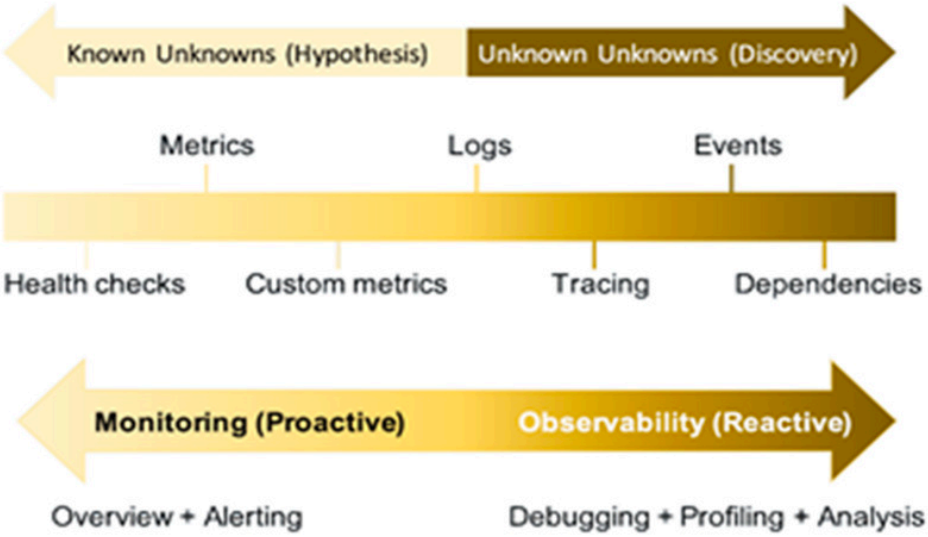


Figure 30 Comparison between monitoring and observability.

3.2.2.1.4 Framework

In this section the proposed monitoring, observability and analytics framework for B5G platforms, shown in Figure 31, is presented. The section is divided into two: the former subsection focuses on the monitoring and observability part, while the latter focuses on the analytics applied on the collected data.

3.2.2.1.5 Monitoring and observability

First, it is crucial to measure the power and/or energy consumption of 5G/B5G components (i.e., containers or virtual machines) to then be able to optimize it. This is a non-trivial task: traditionally the CPU utilization, being CPU the most used hardware component, is considered to evaluate the energy/power consumption of virtualized components [21]. However, it is of paramount importance to consider other resources (e.g., memory and networking) utilization and actual power/energy consumption of containers or virtual machines; this requires a more comprehensive approach.

Our framework is based on the Kubernetes platform, since B5G components (i.e., NFs) are usually deployed as containers and Kubernetes is one of the most well-known platforms for managing containers. Nevertheless, this framework is quite modular and hence it can be

modified in the monitoring, observability and analytics parts to achieve higher orders of scalability and flexibility. In the following, we consider a simplified scenario: we do not actually deploy any B5G networks, but simply containerized applications. As shown in the blue encased part of Figure 31, the monitoring and observability part revolves around the data collection and minor data manipulation. The data is collected from two sources: the Kubernetes platform and the server machine itself. The data collection relies on well-known opensource applications which expose resource utilization and energy/power consumption of either processes (containers included) or of the physical server.

Regarding Kubernetes, two exporters are used: cAdvisor which provides the resource (i.e., CPU, memory, networking) usage of containers and Kube State Metrics (KSM) [which listens to the Kubernetes API server and generates metrics about the state of the objects (e.g., nodes, pods).

Regarding the energy/power consumption, three applications are used: Kepler, Scaphandre and the Raritan exporter. Kepler provides the energy consumption of both the single containers/processes and of the whole servers. Scaphandre provides the same information, but in terms of power consumption. Both Kepler and Scaphandre rely on the information exposed by the Intel RAPL [22] counters.

Moreover, the Raritan exporter provides the active power, temperature and other metrics offered by the Raritan Power Distribution Units (PDUs) to which the servers are plugged. Finally, Node Exporter is used to provide resource (i.e., CPU, memory, networking) usage at the server level. It is worth mentioning that ground-truth data for what concerns power/energy consumption of containers is difficult to obtain. Even exploiting the power consumption of the PDUs (provided by the Raritan exporter) to which the servers are connected is not sufficient, since that includes also environmental variables (e.g., temperature of the room) and all the additional processes executed by the Operating System (OS) itself to keep the server running and to support the virtualized environment. Thus, it is complicated to separate the containers' contribution from the actual power drawn from the PDUs.

All the collected data is stored in a Prometheus database. Prometheus is a time series database, that collects metrics enriched with a timestamp (i.e., representing the time instant in which Prometheus receives said metrics) and with labels which allow for better identification of the data. Prometheus collects information on a pull basis: it polls exporters every scraping interval (herein 10 seconds). Moreover, it provides metrics in four different data types: *i*) counter, that is a cumulative value which can only increase; *ii*) gauge, which is a value able to both decrease and increase; *iii*) histogram, which counts observations in configurable buckets; and *iv*) summary, like histogram samples observations, also provides a total count of observations and a sum of all observed values.

Finally, the metrics saved on Prometheus are accessible either through its Graphical User Interface (GUI) or through its API. The Metric Scraper block shown in Figure F2 has the goal of scraping metrics from Prometheus and then add additional labels (e.g., pod name, namespace and so on obtained by KSM) to the Scaphandre ones.

3.2.2.1.6 Analytics

The analytics part of the proposed framework is shown in the red rectangle in

Figure 31. Our paper delves into the statistical dependence of timeseries, aiming to explore the correlation between resource utilization, power/energy consumption metrics, gathered from different monitoring tools and the actual metrics. However, it is worth highlighting that this is only one possible use of the collected data, and any analytics algorithm could be applied to the Prometheus-like metrics.

The concept of statistical dependence is used to describe a relation between any two features under study, such as individuals, objects, or environmental features. In the absence of deterministic dependence and independence, the statistical dependence remains as the characteristic trait of both features, showing the differences in the distributions of one variable for at least some of the levels of the other [23].

Statistical dependence and independence are fundamental concepts in statistics and machine learning and play a crucial role in various aspects of data analysis and machine learning tasks such as: feature selection [24], anomaly detection [25] [26], dimensionality reduction [27], clustering [24] and optimization [28]. The two use-cases of statical dependence of interest in this article are feature reduction and anomaly detection. The former consists in identifying a subset of available features that is the most relevant for the data analysis of interest. In particular, herein, our goal is to choose the most appropriate application among Kepler and Scaphandre to measure the power/energy consumption. The latter, anomaly detection, consists in finding patterns in data that do not show an expected behaviour and then trigger a warning/alert.

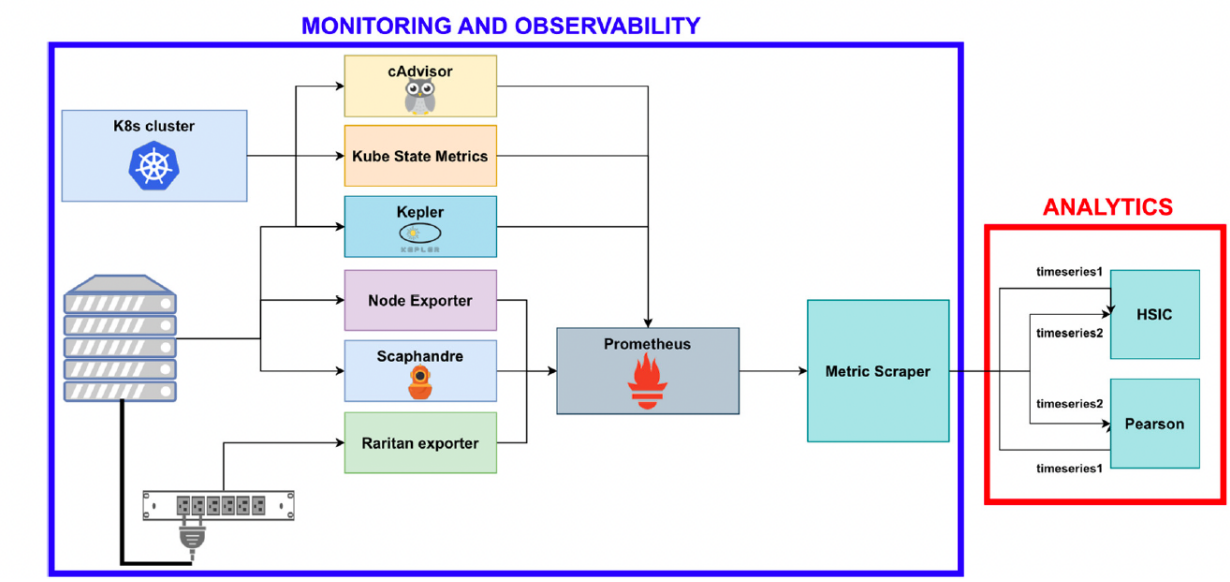


Figure 31 Monitoring, observability and analytics framework.

3.2.2.1.7 Framework tools

The following section is a summary of the tools used in the framework proposed and developed by CNIT. Considering the idea behind the ePEM components (also called NFVCL – NFV Convergence Layer – to highlight the NFV compliant definition), it is possible to use third-party components in an easy way. It is necessary to define only a blueprint connector to allow the management of the external tool. In the following, we consider, at first, the tools with a ready blueprint connector developed, and, later, the ones without blueprint but easy to integrate in the HORSE – ePEM ecosystem.

3.2.2.1.7.1 With Blueprint connector ready

- **Prometheus** (https://prometheus.io) is a free software application used for event monitoring and alerting[8]. It records metrics in a time series database (allowing for high

[8] *https://prometheus.io/docs/introduction/overview*

dimensionality) built using an HTTP pull model, with flexible queries and real-time alerting [29] [30]. The project is written in Go and licensed under the Apache 2 License, with source code available on GitHub (https://github.com/prometheus) and is a graduated project of the Cloud Native Computing Foundation, along with Kubernetes and Envoy[9].

- **Scaphandre** [skafɑ̃dʁ][10] is a metrology agent dedicated to electric power and energy consumption metrics. The goal of the project is to permit to any company or individual to measure the power consumption of its tech services and get this data in a convenient form, sending it through any monitoring or data analysis toolchain.
- **Raritan**[11] began developing KVM switches for IT professionals to manage servers remotely in 1985. Today, as a brand of Legrand, we are a leading provider of intelligent rack PDUs. Our PDUs and other intelligent power and energy monitoring products are transforming how companies manage their data center power chains. Our solutions increase the reliability and intelligence of data centers in 9 of the top 10 Fortune 500 technology companies. Learn more about Legrand at Legrand.us.
- **Filebeat**[12] is lightweight shipper for logs. Whether you're collecting from security devices, cloud, containers, hosts, or OT, Filebeat helps you keep the simple things simple by offering a lightweight way to forward and centralize logs and files.
- **Grafana**[13] is a multi-platform open source analytics and interactive visualization web application. It can produce charts, graphs, and alerts for the web when connected to supported data sources.
- **Loki**[14] is a horizontally-scalable, highly-available, multi-tenant log aggregation system inspired by Prometheus. It is designed to be very cost effective and easy to operate. It does not index the contents of the logs, but rather a set of labels for each log stream. Compared to other log aggregation systems, it:
    - does not do full text indexing on logs. By storing compressed, unstructured logs and only indexing metadata, Loki is simpler to operate and cheaper to run;
    - indexes and groups log streams using the same labels you're already using with Prometheus, enabling you to seamlessly switch between metrics and logs using the same labels that you're already using with Prometheus;
    - is an especially good fit for storing Kubernetes Pod logs. Metadata such as Pod labels is automatically scraped and indexed.
    - has native support in Grafana (needs Grafana v6.0).

  A Loki-based logging stack consists of 3 components:

    - *promtail* is the agent, responsible for gathering logs and sending them to Loki.
    - *loki* is the main server, responsible for storing logs and processing queries.
    - *Grafana* for querying and displaying the logs.

  Loki is like Prometheus, but for logs: it prefers a multidimensional label-based approach to indexing, and want a single-binary, easy to operate system with no dependencies.

[9] *https://www.cncf.io/announcement/2018/08/09/prometheus-graduates*

[10] *https://github.com/hubblo-org/scaphandre*

[11] *https://www.raritan.com*

[12] *https://www.elastic.co/beats/filebeat*

[13] *https://grafana.com*

[14] *https://grafana.com/oss/loki*

Loki differs from Prometheus by focusing on logs instead of metrics, and delivering logs via push, instead of pull.

- **OpenTelemetry**[15] is a collection of APIs, SDKs, and tools. Use it to instrument, generate, collect, and export telemetry data (metrics, logs, and traces) to help you analyze your software's performance and behaviour.

  The **OpenTelemetry Collector** [16] offers a vendor-agnostic implementation on how to receive, process and export telemetry data. In addition, it removes the need to run, operate and maintain multiple agents/collectors to support open-source telemetry data formats (e.g. Jaeger, Prometheus, etc.) to multiple open-source or commercial backends.

3.2.2.1.7.2 Without Blueprint connector ready

- **Elasticsearch**[17] is a distributed, RESTful search and analytics engine capable of addressing a growing number of use cases. As the heart of the Elastic Stack, it centrally stores your data for lightning-fast search, fine-tuned relevancy, and powerful analytics that scale with ease.
- **Logstash**[18] is a free and open server-side data processing pipeline that ingests data from a multitude of sources, transforms it, and then sends it to your favorite "stash."
- **Falco**[19] is a cloud-native security tool designed for Linux systems. It employs custom rules on kernel events, which are enriched with container and Kubernetes metadata, to provide real-time alerts. Falco helps you gain visibility into abnormal behavior, potential security threats, and compliance violations, contributing to comprehensive runtime security.
- **Fluent Bit**[20] is a super-fast, lightweight, and highly scalable logging and metrics processor and forwarder. It is the preferred choice for cloud and containerized environments. It provides robust, lightweight, and portable architecture for high throughput with low CPU and memory usage from any data source to any destination.
- **Jaeger**[21] is an open source, distributed tracing platform for monitoring and troubleshooting workflows in complex distributed systems. Jaeger maps the flow of requests and data as they traverse a distributed system. These requests may make calls to multiple services, which may introduce their own delays or errors. Jaeger connects the dots between these disparate components, helping to identify performance bottlenecks, troubleshoot errors, and improve overall application reliability. Jaeger is 100% open source, cloud native, and infinitely scalable.
- Originally built at ***Lyft***, **Envoy**[22] is a high performance C++ distributed proxy designed for single services and applications, as well as a communication bus and "universal data plane" designed for large microservice "service mesh" architectures. Built on the learnings of solutions such as NGINX, HAProxy, hardware load balancers, and cloud

---

[15] *https://opentelemetry.io*

[16] *https://github.com/open-telemetry/opentelemetry-collector*

[17] *https://www.elastic.co/elasticsearch*

[18] *https://www.elastic.co/logstash*

[19] *https://falco.org*

[20] *https://fluentbit.io*

[21] *(https://www.jaegertracing.io):*

[22] *https://www.envoyproxy.io*

load balancers, Envoy runs alongside every application and abstracts the network by providing common features in a platform-agnostic manner. When all service traffic in an infrastructure flow via an Envoy mesh, it becomes easy to visualize problem areas via consistent observability, tune overall performance, and add substrate features in a single place.

- **Wazuh**[23] provides analysts real-time correlation and context. Active responses are granular, encompassing on-device remediation so endpoints are kept clean and operational. The *Wazuh Security Information and Event Management* (SIEM) solution provides monitoring, detection, and alerting of security events and incidents.

3.2.2.2 Telemetry and tools in the UMU testbed

To understand the results of the tests that will be implemented to validate the proper behavior of the developed HORSE functionalities, some assessment process is needed. To accomplish this, the infrastructure shown in Figure 21 Gaia 5G connectivity schema, has incorporated a variety of telemetry and data collection tools. Information is gathered from different components in the network and the data is stored in time series databases that will facilitate a later analysis. The telemetry technologies are the followings:

- **Netdata**: is a monitoring tool designed to measure the performance of servers and IoT devices. It gathers metrics like CPU, memory, and disk usage. It is known for its efficiency and low impact in the performance of the system. It is also compatible with a variety of operating systems, making this technology flexible and useful in different situations. In addition, this tool offers easy integration with different databases like Prometheus, which is described below.
- **Collectd**: this tool can gather data from Linux and Unix systems. Its main functionality is to compile detailed metrics like CPU and memory usage, networking information, and the status of the storage. This technology it is also very efficient and uses few resources of the system. Collectd is highly configurable and presents a detailed view of the status of the system.
- **Prometheus**: this monitoring platform is particularly useful in distributed environments. It is known for its pull method of gathering the data. The metrics end up exposed, and all the metrics from the different systems can be seen. This way of working allows Prometheus to be efficient and scalable.
- **Grafana**: is just a visualization platform where custom dashboards can be created. This technology stands out for its easy integration with tools like Prometheus. Grafana has an accessible and intuitive interface that allows the creation of dashboards where all the metrics can be checked. This system allows for feedback on the measured process.
- **ELK Stack**: it is a complete platform for the management of logs and data, composed of Elasticsearch, Logstash, and Kibana:
  - **Elasticsearch**: is a search and analytics engine. This tool allows for an efficient storage of metrics and logs and is capable of quickly search through its entire database thanks to its implementation.
  - **Logstash**: is a server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it, for example, to Elasticsearch.
  - **Kibana:** allows users to visualize data in charts and graphs with Elasticsearch.
- **Python**: is a versatile and powerful programming language, known for its clear and readable syntax. Python stands out for its ease of learning, making it a popular choice for beginners as well as an efficient tool for professionals. Using this programming language, different scripts have been developed to help with the gathering of the metrics and to ensure that the telemetry pipeline works smoothly.

With the integration of these tools, the diagram of the testbed is shown in Figure 32, where it can be verified that Netdata has been deployed in all the servers available to control its status.

[23] https://wazuh.com

Netdata, as mentioned previously will provide information like CPU and memory usage, disk status and a variety of networking metrics like latency or throughput. Similar information will be gathered from Collectd, but from network equipment, like routers and switches. The data collected will go to Prometheus where it will be stored and using Grafana it will be visualized. This infrastructure also collects logs from the 5G Core and for this, the ELK Stack has been deployed.

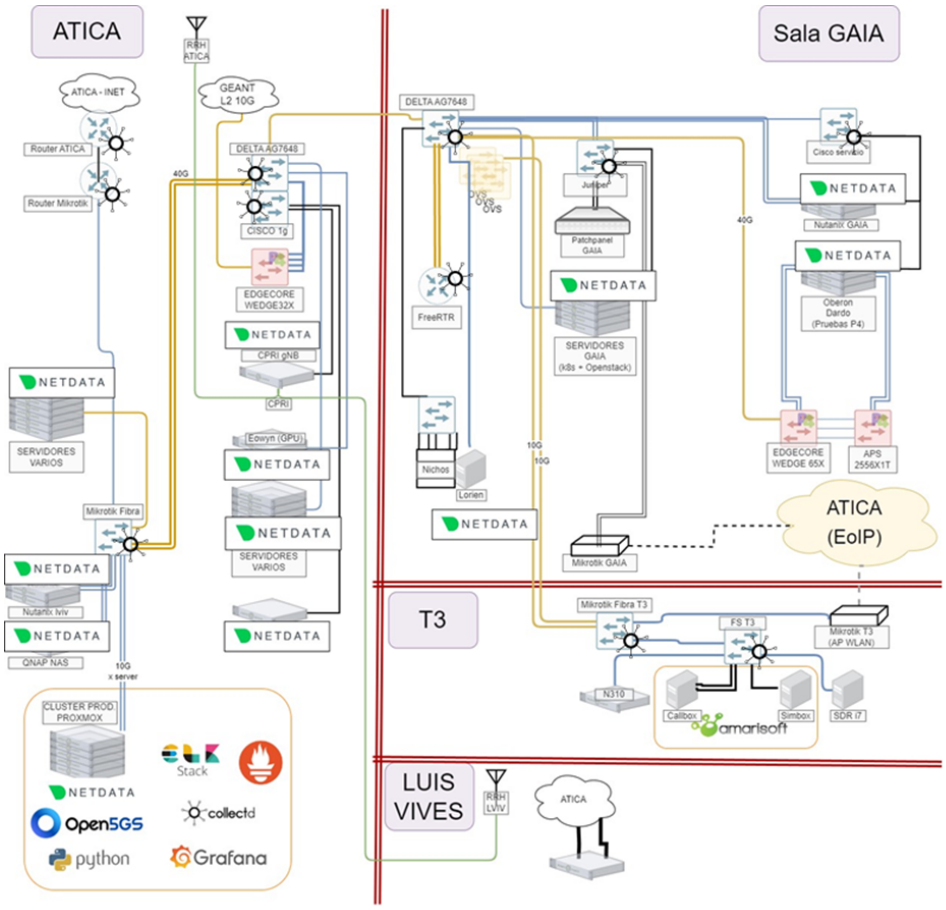


Figure 32 Telemetry tools incorporated

3.2.2.2.1 Traffic Generation

In UMU Testbed, it was developeded a 5G traffic generator aimed to create an efficient utilization of virtual UEs through the Amarisoft's SIMBox. This system leverages a preconfigured VLAN inside the testbed to redirect user's traffic to the UEs. This process is done applying a dynamic solution, allowing an efficient management. The flexibility of this approach can be seen in the capability that the system must activate or to deactivate UEs on demand.

The implementation of this solution enables traffic generation not only in the control plane, but in the user plane as well. Furthermore, this process can be achieved even with devices that are not compatible with 5G. To be able to this, the Amarisoft's SIMbox is used strategically as a bridge between non-5G devices and the 5G network. Doing so, the generated traffic can be captured and analyzed by the different nodes in the network, such as the AMF, that can be found in the N2 interface o UPF, that is in the N3 interface.

In addition, this solution has implemented a capability to dynamically manage virtual UEs. With this, the network is provided with an exceptional versatility since it can adapt to different scenarios where different requirements can be found. Moreover, since the infrastructure is

based on VLAN, it facilitates the management and the routing of the traffic, allowing precise management over the network and the capturing of the data generated by the UEs.

Summarizing, this 5G traffic generator can simulate users in a 5G network while maintaining a flexibility and scalability not only in the control plane, but in the user plane, providing an integral solution for testbeds and test environments in the scope of 5G.

3.2.2.2.2 Signal Measuring

A precise measuring of the signal is needed to assure an optimal performance. In this scope, in the UMU Testbed we have different tools that are essential in this process. From spectrum analyzers to network behavior analysis including a metrics' gathering system. Regarding specific tools and technologies:

- Anritsu's Field Master Pro MS2090A: real time spectrum analyzer delivers performance. With continuous frequency coverage from 9 kHz to 54 GHz, the Field Master Pro MS2090A is specifically designed to meet the test challenges of a full range of other wireless technologies in use today, including: 5G, LTE, wireless backhaul, aerospace/defense, satellite systems, and radar. The Field Master Pro MS2090A delivers the highest levels of RF performance available in a handheld, touchscreen spectrum analyzer, with a displayed average noise level (DANL) of -164 dBm and Third Order Intercept (TOI) of +20 dBm (typical). This makes measurements such as spectrum clearing, radio alignment, harmonic, and distortion even more accurate than previously possible. For modulation measurements on digital systems, 110 MHz modulation bandwidth coupled with best-in-class phase noise performance maximizes measurement precision, while ±0.5 dB typical amplitude accuracy provides confidence when testing transmitter power and spurious.
- Keysight NEMO Handy L2-7 5G engineering phone: Android application that enables measuring wireless diagnostics information of air interface and mobile application quality-of-service (QoS) and quality-of-experience (QoE). This tool is compatible with the latest Android devices, and it offers real-time voice quality testing using POLQA v2 and v3 or ViSQOL. The tool allows for forcing commands related to system, band, carrier lock, SC, channel, and PCI lock for selected devices. Large-scale site acceptance projects can be conducted with the optional Nemo Handy Site Acceptance Edition and Nemo Cloud. It supports the Network Performance Score (NPS), a single metric for assessing network Quality of Experience (QoE) based on existing KPIs defined by ETSI and ITU-T. Additionally, a navigation application provides turn-by-turn instructions for drive test campaigns, and it allows combining scanning and indoor measurements using portable, battery-operated devices such as PCTEL Gflex/HBflex/IBflex and R&S®TSMA6 Autonomous Mobile Network scanner devices.
- System for obtaining real-time status of the OBU's own modem: in the scope of 5G networks, On-Board Units (OBUs) perform an important part in guaranteeing connectivity between vehicles and devices. Taking this premise into account, to have a system with an optimal performance and with an efficient network management, we have developed a complex system that can collect metrics from the status of the modem in the OBU. This system implements different monitoring strategies like a continuous observability.

3.2.2.2.3 Robot Framework

Robot Framework is a generic open-source automation framework. It can be used for test automation and robotic process automation (RPA). Robot Framework is supported by Robot Framework Foundation. Many industry-leading companies use the tool in their software development. Robot Framework is open and extensible. Robot Framework can be integrated with virtually any other tool to create powerful and flexible automation solutions. Robot Framework is free to use without licensing costs. Robot Framework has an easy syntax, utilizing human-readable keywords. Its capabilities can be extended by libraries implemented

with Python, Java or many other programming languages. Robot Framework has a rich ecosystem around it, consisting of libraries and tools that are developed as separate projects.

Robot Framework could significantly improve the presented CI/CD pipeline by providing test automation. This framework can integrate acceptance and regression tests into the pipeline to guarantee a thorough validation of the software. This process will enhance the detection of problems, reducing the chances of making faulty deployments. Furthermore, Robot Framework's flexibility allows integration with a variety of technologies and the execution of test in different environments. Moreover, Robot will display detailed reports where the results are highlighted. These reports will help the identification of problems and their swift correction. Combining this framework with GitHub Actions will create a complete, efficient solution that speeds up the development cycle.

3.2.2.3 Tools in the UPC testbed

The tools that will be used in the UPC 5G testbed are listed below.

- **Open5GCore**: This tool, created for research, testing, and development in the field of 5G networks, allows the simulation of the core of 5G networks with all its functionalities, which is important for testing and validating various aspects of this emerging technology, from mobility management to the provisioning of high-speed services. Open5GCore can be integrated with other open-source components and tools to create a complete simulation and testing environment. It can be easily integrated with the User Equipment Radio Access Simulator (UERANSIM).
- **UERANSIM**: This software provides a realistic simulation of radio access in 5G networks. It brings the functionalities of antenna (gNodeB) and user equipment (UE), allowing the recreation of connectivity and mobility scenarios across different areas. When integrated with Open5GCore, UERASIM completes the necessary infrastructure to perform end-to-end 5G network tests, from the radio access layer to the core network.
- **Kubernetes**: It is a container orchestration platform that allows efficient management and deployment of different components of the network infrastructure. In the specific case of this environment, Kubernetes manages the deployment of containers containing related applications and services, which are orchestrated in a Private Cloud. Additionally, Kubernetes can run monitoring and visualization tools like Grafana, Prometheus, and Loki to provide insights into everything happening in the network.
- **Docker**: It is a platform for creating applications in containers, orchestrated by Kubernetes. Docker facilitates the portability and scalability of services required for the network infrastructure. Docker containers contain applications and their dependencies, allowing them to run independently of the underlying operating system. This greatly simplifies the deployment of network components in heterogeneous environments and facilitates resource management.
- Prometheus: This software allows performance and availability monitoring of applications and network infrastructure, as well as the generation of alerts. When integrated with Kubernetes, Prometheus can collect metrics and logs from all components deployed in the environment (including pods), providing complete visibility into the system's status and enabling early detection of potential issues.
- **Grafana**: It is an analytics and visualization platform that complements Prometheus by providing an intuitive graphical interface for exploring and visualizing real-time monitoring data. Implemented in Kubernetes, Grafana seamlessly integrates with Prometheus and Loki, allowing comprehensive monitoring and effective visualization of network metrics and logs.

- **Loki**: This log storage solution integrates closely with Prometheus and Grafana to provide efficient log management in container environments. Implemented in Kubernetes, Loki provides a scalable and cost-effective solution for storing and querying large volumes of network logs, supporting real-time analysis and issue resolution.

3.3 Planning for verification and validation campaigns

The verification and validations campaigns are aligned with the HORSE roadmap, taking in the consideration the following major releases:

- IT (Intermediate)-1 version: consists in the first HORSE release. Planned to be released at the end of June/2024 (M18)
- IT(Intermediate)-2 version: consists in the second HORSE release – Planned to be release at the end of August/2025 (M32)
- Final version: Final HORSE release- Planned to be released at the end of the project (December/2025 – M36)

Next Figure 33 illustrates the timeline for HORSE releases showing the estimated validation periods. It was defined two main validation phases for the overall HORSE roadmap.

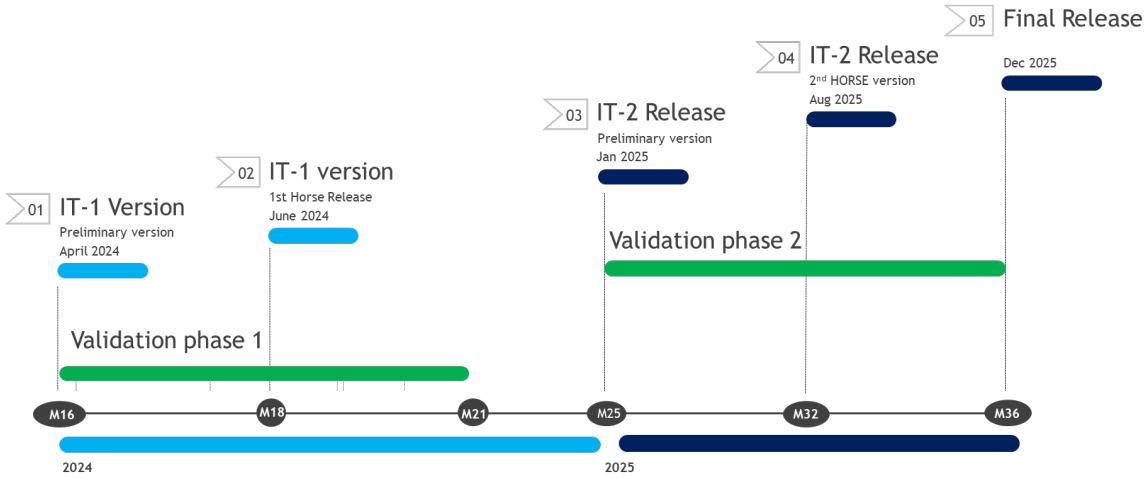


Figure 33 Timeline for HORSE releases

It is important to mention that for each IT version will be released a preliminary version, as illustrated in Figure 33, allowing to start the unitary test, the integration, the setup in the defined testbed and to implement some refinements in the software according with the preliminary test results.

The goal will be to report the achievements and the results, for each validation planned campaigns.

Specifically for the first HORSE release, and according with the progress of the technical implementation, partners agreed (plenary meetings and WP5 bi-weekly meetings) a calendar, depicted in Figure 34 with a set of software preliminary releases, test campaigns for the production of the HORSE release v1.0 (including intermediate milestones and meeting).

The main goals and milestones are:

- Conclusion of software HORSE modules, in the scope of WP3 and WP4
- Release preliminary version v.01 and upload the software at common Git-Hub

- Release preliminary version v0.2, refine the technical demonstrations and setup the tools, such as traffic generators and dataset capture
- Deployment of Version v02 in the testbeds
- Final setup of testbeds, including the 5Gnet topology
- Release and deployment in the testbed of official HORSE v1.0
- Technical demos integration and validation
- Release v1.1 (bug fixing)

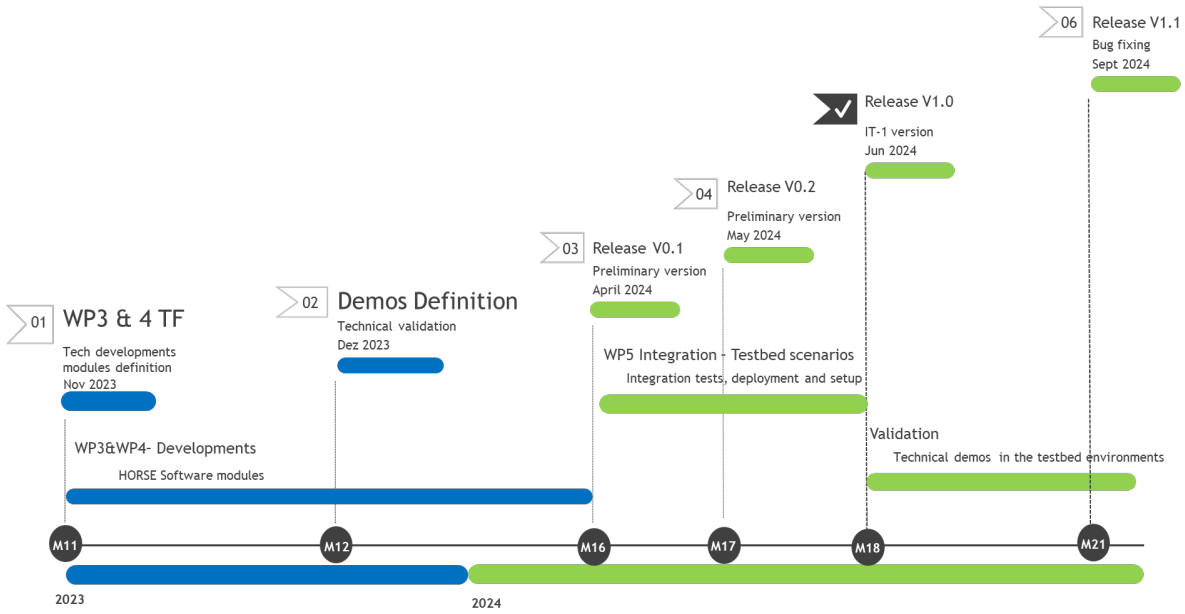


Figure 34 Timeline for HORSE IT-1 version

3.3.1 Integration Tests

The goal of this step is to implement and report the Integration tests performed for the HORSE releases.

The following integration tests in pairs are planned (combination 2 or more modules). Each pre-integration tests verify the input/output, data model, exchanged parameters and basic functionality per module. This will be executed in the virtualized integration infrastructure provided by UMU.

- DEME and DTE
- DTE, IBI and EM (incl common database)
- IBI and RTR
- SAN (NDTs) and IBI interfaces
- SM and Preproc [and datasets for DEME]
- ePEM and Testbeds
- SM and Testbeds

3.3.2 Overall HORSE Demo Tests planned

At the moment of the deliverable edition, several technical demonstrations are planned. Next Table 9 illustrates a very preliminary list of the demonstrations for HORSE release IT-1 and its mapping on the associated testbeds in which will be executed. This table also indicates the type of attack considered at the technical validation.

Phase	*Demonstration and Modules*	*Attack and Testbed*
DETECTION	Demo 1 is a DEME centric combined with smart monitoring & preprocessing modules providing captured dataset (incl. data traffic and attacks) and output towards DTE	Attack is NTP DDoS over CNIT testbed. Dependence with the data generation & collection
ANALYSIS	Demo 2 realises the analysis using a network digital twin. It involves modules such as IBI, EM and NDT (centric demo) to decide the mitigation actions	Attack considered is DoS DNS Output is the decision to apply mitigation action after the NDT analysis. Environment NDT_1
ENFORCEMENT	Demo 3 shows the action enforcement from intent IBI towards chain → RTR → ePEM and DOC	Attack DDoS DNS Over UMU testbed. Explore possible multidomain enforcement demonstration
PREDICTION	Demo 4 is NDT centric for the prediction of an attack and involves SM, Preproc and PAG modules	Type of predicted attack (TBC) and it will depend on possible monitoring over certain NFs. Environment NDT_2.
DETECTION	Demo 5 is around DTE recommender with certain simulation data as collected from open5GS NFs	Attack on signaling PFCP traffic and collected data from open5GS. To assess whether UPC testbed is possible.

Table 9 Planning for demonstrations and validation

4 Conclusions

In conclusion, this deliverable describes the demonstration strategy focused on the first HORSE Release, identifying the modules being developed, in the scope of WP3 and WP4, according with a well-defined architecture, achieved in the scope of WP2.

To reach this objective, incremental system integration approach is followed, and several technical demonstrations scenarios are defined, taking into consideration the HORSE modules, and three testbeds are being prepared. A considerable effort has been performed to select and configure specific tools, to support all the testbeds as well as the validations and a roadmap related to validation campaigns is presented.

Since this is the first deliverable of WP5, further improvements and updates for the demonstration strategy are expected while new HORSE platform Releases are being deployed. Those outputs will be integrated on the future planned deliverables (D5.2, D5.3 and D5.4).

This will guide to a continuous validation process applicable to the HORSE releases assuring the quality and the maturity of the achievements, based on acceptance tests to validate the HORSE features and to measure the overall performance according to well-defined KPIs.

5 References

[1] D2.2 HORSE Architectural Design V1.0 (IT-1), available:HORSE shared repository.

[2] D2.1 HORSE Landscape: Technologies, state of the art, AI policies and requirements, available: HORSE shared repository.

[3] Zhao, Serebrenik, Zhou, Filkov and Vasilescu, "32nd IEEE/ACM International Conference on Automated Software Engineering (ASE).," in *The impact of continuous integration on other software development practices: A large-scale empirical study.https://doi.org/10.1109/ase.2017.8115619*, 2017.

[4] M.Virmani, “Fifth International Conference on the Innovative Computing Technology (INTECH 2015),,” in *"Understanding DevOps & bridging the gap from continuous integration to continuous delivery",pp. 78-82, 10.1109/INTECH.2015.7173368 Keywords:{Software;Organizations;Testing;Automation;Optimization;Production;DevOps;Continuous Integration;Continuous Delive*, Galcia, 2015.

[5] "Rodríguez, P., Haghighatkhah, A., Lwakatare, L. E., Teppola, S., Suomalainen, T., Eskeli, J., ... & Oivo, M. (2017). Continuous deployment of software intensive products and services: A systematic mapping study. Journal of systems and software, 123, 263-2".

[6] "Atlassian. (n.d.). What is continuous integration. Atlassian. Retrieved September 8, 2022, from https://www.atlassian.com/continuous-delivery/continuous-integration," [Online].

[7] D3.1 - HORSE Platform Intelligence developed (IT-1)available: HORSE shared repository.

[8] D4.1-HORSE AI-assisted human-centric Secure and Trustable Orchestration, available:Horse shared repository.

[9] ""“Slices-SC - Scientific LargeScale Infrastructure for Computing/ Communication Experimental Studies.” .," [Online]. Available: https://slices-sc.eu/..

[10] ""“ESFRI - European Strategy Forum on Research Infrastructures.”," [Online]. Available: https://www.esfri.eu.

[11] "P4 Language Consortium,," [Online]. Available: https://p4.org/..

[12] "HUAWEI 5G CPE Pro 2.," [Online]. Available: https://consumer.huawei.com/en/routers/5g-cpe-pro-2/.

[13] "AMARI UE Simbox Series,," [Online]. Available: https://www.amarisoft.com/products/test-measurements/amari-ue-simbox/.

[14] "Apache Guacamole,," [Online]. Available: https://guacamole.apache.org/.

[15] "SCRCPY: An Android Screen Mirroring Tool,," [Online]. Available: https://scrcpy.org/.

[16] "GARR - the italian research and education network,," [Online]. Available: https://www.garr.it/en/.

[17] "R. Bolla et al., "A Multi-Tenant System for 5/6G Testbed as-a-Service," 2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS), Bangalore, India, 2023, pp. 768-773, doi: 10.1109/COMSNETS56262.2023.10041360.".

[18] "How to Monitor Service: The Fundamental Framework.," [Online]. Available: https://hackernoon.com/service-monitoring-the-conceptualframework-2ebffebb362d.

[19] "M. Usman, S. Ferlin, A. Brunstorm, and J. Taheri "A Survey on Observability of Distributed Edge & Container-Based Microservices," in IEEE Access, vol. 10, pp. 86904–86919, July. 2022, doi: 10.1109/ACCESS.2022.3193102.".

[20] "IBM, "The Enterprise Guide to Observability," [Online]. Available: https://www.ibm.com/resources/automate/observability/.

[21] "X. Zhang, Z. Shen, B. Xia, Z. Liu and Y. Li, "Estimating Power Consumption of Containers and Virtual Machines in Data Centers," 2020 IEEE International Conference on Cluster Computing (CLUSTER), Kobe, Japan, 2020, pp. 288-293, doi: 10.1109/CLUSTER49012.20".

[22] "KN. Khan, M. Hirki, T. Niemi, JK, Nurminen, B. Schölkopf, and Z. Ou "RAPL in Action: Experiences in Using RAPL for Power Measurements," in ACM Transactions on Modeling and Performance Evaluation of Computing Systems, vol. 3, issue. 2, pp. 1–26, Jan. 2023,".

[23] P. Armitage, and T. Colton, "Encyclopedia of biostatics," Wiley Press, 2005, PP. 5606-5609,SBN: 9780470011812, doi:10.1002/0470011815..

[24] T. Wang, X. Dai, and Y. Liu, "Learning with Hilbert–Schmidt independence criterion: A review and new perspectives," in Knowledge-Based Systems, vol. 234, Dec. 2021, doi: 10.1016/j.knosys.2021.107567..

[25] "W. Huo, W. Wang, and W. Li, "AnomalyDetect: An Online Distance-Based Anomaly Detection Algorithm," in 26th International Conference on Web Services, pp. 63–79, June. 2019, doi: 10.1007/978-3-030-23499-7_5.".

[26] C. Zhang, W. Zuo, P. Yang, Y. Li, and X. Wang "Outsourced privacy preserving anomaly detection in time series of multi-party," in China Communications, vol. 19, Issue. 2, pp. 201–213, Feb. 2022, doi: 10.23919/JCC.2022.02.016..

[27] K. Chen, "Indirect PCA Dimensionality Reduction Based Machine Learning Algorithms for Power System Transient Stability Assessment," in IEEE Innovative Smart Grid Technologies – Asia (ISGT Asia), May. 2019, doi: 10.1109/ISGT-Asia.2019.8881370..

[28] "B. Guenin, J. Könemann, and J. Tunçel, "A Gentle Introduction to Optimization," Cambridge University Press, July. 2014, doi:10.1017/CBO9781107282094.".

[29] "James Turnbull, “Monitoring with Prometheus,” Turnbull Press, June, 2018, isbn:978-0-9888202-8-9.".

[30] Prometheus, "Prometheus: From metrics to insight. Power your metrics and alerting with a leading open-source monitoring solution," prometheus.io, Retrieved December 26, 2018..

[31] "Canonical MaaS,," [Online]. Available: https://maas.io/.

[32] "RedHat Ansible - Drive automation across open hybrid cloud deployments,," [Online]. Available: https://www.ansible.com/.

[33] “Prometheus,,” [Online]. Available: https://docs.openstack.org/openstack-helm-infra/latest/monitoring/prometheus.html..

[34] "Elastic Stack - Meet the search platform that helps you search, solve, and succeed,," [Online]. Available: https://www.elastic.co/elastic-stack..

[35] “ETSI, ETSI GS NFV-SOL 006 V2.7.1, “Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on YANG Specification”, December, 2019”.