Ana Luísa
Gonçalves da Silva
Marques Ferreira

Testbed para Aplicações e Serviços 5G

5G Application & Service Testbed

Universidade de Aveiro
2023

Ana Luísa Gonçalves da Silva Marques Ferreira

Testbed para Aplicações e Serviços 5G

5G Application & Service Testbed

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Diogo Gomes, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Rui Luís Andrade Aguiar, Professor catedrático do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho aos meus pais.

o júri / the jury

presidente / president

Arnaldo Silva Rodrigues de Oliveira
Professor Associado na Universidade de Aveiro

vogais / examiners committee

Ana Cristina Costa Aguiar
Professora Auxiliar na Universidade do Porto

Diogo Nuno Pereira Gomes
Professor Auxiliar na Universidade de Aveiro

agradecimentos / acknowledgements

Começo por agradecer aos meus pais e avós pelo incansável apoio, pela orientação ao longo da minha vida e por me oferecerem condições para ser bem sucedida tanto a nível profissional como pessoal. Quero também agradecer ao meu namorado, Pompeu Costa, por me apoiar nos momentos mais difíceis e incentivar a nunca desistir. Agradeço também aos meus amigos, nomeadamente ao João Gameiro, ao Marco Ramos e por fim, mas não menos importante ao Rodrigo Martins, que me acompanharam nestes últimos 5 anos e que passaram tanto momentos incriveis como difíceis comigo. De seguida, quero agradecer aos meus orientadores, em especial, ao Professor Doutor Diogo Gomes por me ter ajudado e orientado neste percurso. Quero também deixar um agradecimento ao Rafael Direito e ao Daniel Gomes por me terem ajudado e acompanhado o meu trabalho de perto no Instituto de Telecomunicações. Deixo também os meus agradecimentos aos meus restantes colegas do Instituto de Telecomunicações.

Palavras Chave 5G, NFV, Network Application, Validação, NEF.

Resumo

Com o crescimento das tecnologias 5G, há a necessidade de adotar o paradigma NFV para responder às necessidades de um ambiente 5G, o que leva à separação do hardware das funções que ele fornece, acelerando o desenvolvimento de serviços. No entanto, a adoção de NFV traz muitos desafios, um dos quais é a validação das VNF. O 5GASP é um projeto europeu que visa a encurtar o processo de ideia-ao-mercado, criando um ambiente de teste 5G totalmente automatizado. Em alinhamento com os objetivos do 5GASP, esta dissertação procura criar testes para a validação de Network Applications usadas em ambiente 5G. Além disso, a Arquitetura Baseada em Serviços oferece o NEF, que permite o acesso de aplicações externas às funções da rede 5G. Com estes conceitos em mente, a solução apresentada por esta dissertação utiliza o NEF para verificar se uma Network Application consegue interagir corretamente com uma rede 5G. Isto é alcançado integrando o NEF no pipeline de validação do 5GASP e desenvolvendo testes para avaliar o comportamento das Network Applications. No geral, este documento apresenta a arquitetura definida e a implementação desses mecanismos, juntamente com seus respectivos resultados.

Keywords 5G, NFV, Network Application, Validation, NEF.

Abstract

With 5G technologies growing, there is the need to adopt the NFV paradigm to align with the demands of a 5G environment. This entails the decoupling of hardware from the functions it provides, expediting service development. However, the adoption of NFV raises many challenges, one of which is the validation of Virtualized Network Functions. 5GASP is a European project that aims to shorten the idea-to-market process by creating a fully automated and self-service 5G testbed. In alignment with 5GASP's objectives, this dissertation seeks to design tests for the validation of Network Applications used within a 5G environment. Furthermore, the Service-Based Architecture offers the NEF, which enables external application access to 5G core functions. With these concepts in mind, the solution provided by this dissertation uses NEF to verify whether a Network Application can correctly interact with a 5G Network. This is achieved by integrating NEF into the validation pipeline of 5GASP and developing tests to assess the behavior of Network Applications. In summary, this document presents the designed architecture and the implementation of these mechanisms, along with their respective results and outcomes.

Contents

Contents i

List of Figures v

List of Tables vii

Acronyms ix

1 Introduction 1

1.1 Motivation . . . 2
1.2 Goals . . . 2
1.3 Dissertation Outline . . . 2

2 Background Concepts and Work Proposal 3

2.1 Before 5g . . . 3
2.2 Network Function Virtualization . . . 4
2.2.1 European Telecommunications Standards Institute (ETSI) Management and Orchestration (MANO) . . . 5
2.3 MANO Solutions . . . 6
2.3.1 Open Network Automation Platform (ONAP) . . . 6
2.3.2 Open Source MANO (OSM) . . . 7
2.4 Network Applications . . . 8
2.5 5th Generation Mobile Network (5G) Core System Architecture . . . 9
2.5.1 5G Core Network Capability & Network Exposure Function . . . 11
2.6 Agile Methodology . . . 12
2.6.1 DevOps . . . 12
2.6.2 Continuous Integration (CI)/Continuous Delivery (CD) . . . 13
2.6.3 Harnessing DevOps and CI within a Network Function Virtualization (NFV)Context . . . 14
2.7 Testing Automation Frameworks . . . 15

2.8 Related Work . . . 15
2.8.1 VITAL-5G . . . 16
2.8.2 EVOLVED-5G . . . 20
2.8.3 5GMediaHUB . . . 29
2.8.4 5GASP . . . 32
2.9 Work Proposal . . . 35
2.10 Chapter Summary . . . 36

3 Architecture **39**
3.1 Problem Statement . . . 39
3.2 Architecture and Specifications . . . 41
3.2.1 Components . . . 41
3.2.2 Modules' Interaction . . . 42
3.3 Testing Pipeline . . . 43
3.3.1 Report File . . . 44
3.4 Chapter Summary . . . 45

4 Implementation **47**
4.1 Network Exposure Function (NEF) Emulator . . . 47
4.1.1 Northbound Application Programming Interfaces (APIs) . . . 47
4.1.2 Report Handler . . . 49
4.1.3 Emulation Endpoints . . . 49
4.1.4 Generating Values . . . 49
4.2 Report API . . . 51
4.3 Mini API . . . 51
4.4 Validation Tests . . . 52
4.4.1 Local Test Repository (LTR) . . . 53
4.4.2 5G Readiness . . . 53
4.4.3 Security . . . 55
4.4.4 Generation of Values . . . 56
4.5 Chapter Summary . . . 57

5 Results **59**
5.1 5G Readiness Tests . . . 59
5.1.1 Testing Scenario . . . 59
5.1.2 Results . . . 64
5.2 Security Tests . . . 69
5.2.1 Testing Scenario . . . 69
5.2.2 Results . . . 69

5.3 Generation of Values 70
5.3.1 Testing Scenario 70
5.3.2 Results 71
5.4 Chapter Summary 71

6 Conclusions **73**
6.1 Conclusions 73
6.2 Future Work 74

References **75**

Appendix **79**
Report Handler Function 79
Create Monitoring Subscription Test 81
Test Results Visualization Dashboard (TRVD)'s Web Interface 82
Robot File for the Unsuccessful Execution 83

List of Figures

2.1 NFV Reference Architectural Framework [8] 5
2.2 ONAP's London Release - Platform Architecture Diagram [12] 6
2.3 OSM Main Architectural Components [19] 8
2.4 5G Core System Architecture [27] 11
2.5 DevOps Axes [31] 13
2.6 CI/CD Pipeline [31] 14
2.7 VITAL-5G Testing & Validation Framework [42] 17
2.8 VITAL-5G Testing & Validation Procedure [42] 18
2.9 Methodology of defining metrics in the Vital-5G [47] 20
2.10 EVOLVED-5G System Architecture [50] 21
2.11 Network Application lifecycle phases and relation with architectural components [50] . . 22
2.12 EVOLVED-5G NEF Emulator Architecture [30] 26
2.13 EVOLVED-5G NEF Emulator Swagger User Interface (UI) [54] 27
2.14 EVOLVED-5G NEF Emulator Dashboard [54] 27
2.15 EVOLVED-5G NEF Emulator Map [54] 28
2.16 Network Application tests [57] 29
2.17 5GMediaHUB Logical Architecture [59] 30
2.18 Overall Experimentation Tools Layer Architecture [59] 31
2.19 5G Application & Services Experimentation and Certification Platform (5GASP)'s CI/CD Service High Level Architecture [67] 34
2.20 5GASP Architectural Workflow [67] 34

3.1 High Level Interactions with the Shared file 40
3.2 Components' Interaction 42
3.3 Proposed Architecture 43

4.1 Endpoints added by 5GASP 49
4.2 Developed Handover endpoint 49
4.3 UI for the Generation of Values 50
4.4 Endpoints that enable the Generation of Values 50

4.5 Report API endpoints 51
4.6 Implemented Components' Interaction 52
4.7 Test's Structure 54

1 TRVD's Web Interface Portraiting an Unsuccessful Validation Process 82
2 *report.html* Robot File for the Unsuccessful Execution of the NEF Handovers Test 83

List of Tables

4.1 Northbound APIs 48
4.2 Implemented 5G Readiness Tests 54

5.1 Dummy Network Applications' Communication with NEF Emulator 62
5.2 Dummy Network Application 1 Results 65
5.3 Dummy Network Application 2 Results 65
5.4 Dummy Network Application 3 Results 66
5.5 Dummy Network Application 4 Results 66
5.6 Dummy Network Application 5 Results 67
5.7 Dummy Network Application 6 Results 68
5.8 Dummy Network Application 7 Results 68
5.9 Dummy Network Application 8 Results 69
5.10 Dummy Network Application 9 Results 69
5.11 Dummy Network Application 10 Results 70
5.12 Dummy Network Application 11 Results 70
5.13 Dummy Network Application 12 Communication with NEF Emulator 70
5.14 Dummy Network Application 12 Results 71

Acronyms

3GPP	3rd Generation Partnership Project
5G	5th Generation Mobile Network
5GASP	5G Application & Services Experimentation and Certification Platform
5GC	5G Core
5GPPP	5G Infrastructure Public Private Partnership
AF	Application Function
AGV	Automated Guided Vehicle
AMF	Access and Mobility Management Function
API	Application Programming Interface
AR	Augmented Reality
AUSF	Authentication Server Function
BSS	Business Support Systems
CAPEX	Capital Expenditures
CAPIF	Common API Framework
CI	Continuous Integration
CD	Continuous Delivery
CDSO	Cross-Domain Service Orchestrator
CLI	Command-Line Interface
CNF	Containerized Network Function
CP	Control Plane
CPU	Central Processing Unit
DN	Data Network
ECOMP	Enhanced Control, Orchestration, Management & Policy
ELCM	Experiment Life-Cycle Manage
ETSI	European Telecommunications Standards Institute
HTML	HyperText Markup Language
IM	Information Model
IoT	Internet of Things
IP	Internet Protocol
ISG	Industry Specification Group
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LTR	Local Test Repository
MANO	Management and Orchestration
NAS	Non-Access Stratum
MTP	Motion-to-Photon
NBI	Northbound Interface
NEF	Network Exposure Function
NF	Network Function
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NFVO	Network Function Virtualization Orchestrator
NIDD	Non-IP Data Delivery
NODS	Network Application Onboarding and Deployment Service
NPN	Non-Public Networks
NRF	Network Repository Function
NS	Network Service
NSO	Network Service Orchestrator
NSSAAF	Network Slice Specific Authentication and Authorization Function
NSSF	Network Slice Selection Function
ONAP	Open Network Automation Platform
Open-O	Open Orchestrator
OPEX	Operating Expenditures
OSM	Open Source MANO
OSS	Operations Support Systems
PCF	Policy Control Function
PD	Performance Diagnosis
PDU	Protocol Data Unit
QoE	Quality of Experience
QoS	Quality of Service
RA	Results Analytics
RAM	Random Access Memory
RAN	Radio Access Network
REST	Representational State Transfer
RO	Resource Orchestrator
RSRP	Reference Signal Received Power
SA	Stand Alone
SBA	Service-Based Architecture
SCP	Service Communication Proxy
SDN	Software-Defined Networking
SLA	Service Level Agreement
SME	Small and Medium-sized Enterprises
SMF	Session Management Function

SMS	Short Messaging Service
SUPI	Subscription Permanent Identifier
SSL	Secure Sockets Layer
T&L	Transportation and Logistics
TRVD	Test Results Visualization Dashboard
TSP	Telecommunication Service Provider
UDM	Unified Data Management
UE	User Equipment
UI	User Interface
UP	User Plane
UPF	User Plane Function
VIM	Virtualised Infrastructure Manager
VNF	Virtual Network Function
VR	Virtual Reality
WP	Work Package
XML	Extensible Markup Language
XR	Extended Reality
YAML	YAML Ain't Markup Language
YANG	Yet Another Next Generation

CHAPTER 1

Introduction

In recent years, Telecommunication Service Providers (TSPs) have been undergoing a transition from hardware-centric systems to software-driven ones. This shift arises from the fact that physical equipment often reaches the end of its lifecycle quickly, requiring frequent replacements. Additionally, this hardware is typically designed for specific tasks, making it challenging to adapt for different purposes and functionalities.

With Traditional Networks, TSP struggled with significant Capital Expenditures (CAPEX) and Operating Expenditures (OPEX), due to limitations in flexibility and scalability, which in turn, lead to extended time-to-market periods [1].

Moreover, the adoption of 5G brings higher bandwidth, reduced latency, low power consumption and more cost-effective networks, thus enabling better and faster network services, which is a crucial necessity in today's fast-paced and interconnected world [2].

NFV emerges as a solution to the challenges previously mentioned. The NFV paradigm facilitates the separation of Network Functions (NFs) from the hardware they traditionally rely on, enabling their deployment through Virtual Network Functions (VNFs) [3].

By adopting this approach, TSPs can reduce expenses related to equipment and its maintenance. Additionally, this method offers scalability, flexibility, and time-saving benefits, resulting in increased production efficiency[3].

Nevertheless, transitioning from hardware-based systems to virtualized ones using the NFV paradigm presents numerous challenges.

One such challenge within NFV is the validation of VNFs, which is the focus of this dissertation. Before deploying their Network Applications with real operators, application owners must confirm that their applications function effectively in a 5G environment. Hence, the necessity for detailed validation processes.

To facilitate NFV technology, 3rd Generation Partnership Project (3GPP) has established specifications for the 5G System Architecture. The main distinction from traditional network architectures is the utilization of NFs that provide services to others through the use of APIs. This architecture is referred to as Service-Based Architecture (SBA). One noteworthy NF

is NEF. NEF offers an interface for external applications to access capabilities from the 5G Core (5GC), such as NFs exposed by the NEF. However, there is currently no implementation of a real NEF available. Nonetheless, NEF is an interesting NF for validating VNFs and evaluating their compatibility with the 5G infrastructure. The EVOLVED-5G[1] project has developed a NEF emulator that provides some functionalities and will be used in the context of this dissertation as the foundation for tests validating VNFs.

1.1 Motivation

The 5GASP[2] project represents a Horizon 2020 research and innovation initiative funded by the European Union. Its primary objective is to reduce the idea-to-market timeline by establishing a comprehensive and self-service testbed within Europe. Furthermore, 5GASP is dedicated to fostering innovation in the realm of experiment and test operations across diverse domains. It offers essential software support tools for the secure and reliable CI/CD of VNFs [4]. Given that 5GASP offers a testbed for application owners to validate their applications, the need for having and conducting tests becomes self-evident.

1.2 Goals

The main objective of this dissertation is to create tests that enable the validation of VNFs. Moreover, it is crucial not only to integrate these tests into the 5GASP project but also to enhance the NEF emulator. This extension of the NEF emulator is vital to enable the creation of tests centered around it, adding another goal for this dissertation. The work developed is associated with the 5GASP's Work Package (WP) 5, with the intention of being incorporated into the project.

1.3 Dissertation Outline

The content of this dissertation is structure as follows. Chapter 2 provides an overview of the background concepts and related work relevant to this dissertation. Based on the background concepts covered in Chapter 2, a work proposal is formulated in Section 2.9, aligned with the goals introduced in Section 1.2.

Then, Chapter 3 presents the proposed architecture designed to address the work proposal, integrating the NEF emulator with the 5GASP's Validation Pipeline. Chapter 4 covers the architecture's implementation and the creation of tests for validating VNFs.

Following the presentation of the architecture and implementation, Chapter 5 details how the solution was tested and presents the results obtained from the developed tests.

Chapter 6 provides a conclusion and offers suggestions for future work.

Lastly, this document includes a list of references used to write this dissertation, along with an appendix.

[1] https://evolved-5g.eu/
[2] https://www.5gasp.eu/

CHAPTER 2

Background Concepts and Work Proposal

This Chapter serves as an introduction to all the concepts essential for comprehending the work developed in this dissertation.

To begin, the transition from traditional network systems to the current 5G era is introduced, highlighting the adoption of the NFV paradigm.

A key element of NFV is the MANO Framework, a platform responsible for the management and orchestration of NFV. This Chapter provides an overview of this platform and its several implementations, including ONAP and OSM.

Given the dissertation's focus on the validation of VNFs, the concept of Network Applications is explained, since they are the applications that are deployed as VNFs and need to be validated.

Additionally, it is addressed the 5G Core System Architecture, with particular emphasis on the NEF, as well as the significance of DevOps and CI/CD concepts, which are the key enablers for the tests and validation process that this dissertation aims to create, and even Automation Testing Frameworks are addressed.

Furthermore, various related projects are mentioned to give an overview of other research efforts related to VNFs validation and testing.

Lastly, a work proposal is formulated.

2.1 Before 5G

Over the last decade, there has been a substantial increase in the demands placed on mobile and other communication networks. Initially limited to basic functions like phone calls and text messages, these networks now play a crucial role in supporting the foundation of a modern digital economy. As the world shifts away from 20th century operational paradigms, these networks are expected to facilitate new modes of operation, adapting to the challenges of the 21st [5].

The motivation behind 5G extends beyond creating a new core network, it arises from the convergence of various requirements and demands, including [5]:

1. Expanding business case requirements from a diverse range of economic players, including industrial firms pioneering novel use cases.
2. Emerging core network technologies fostering expectations of increased operational efficiency and flexibility.
3. Changing dynamics in balancing business, societal, and environmental considerations to deliver services in innovative ways.

Furthermore, traditional network devices, designed with specific purposes, lead to increased costs when adopting new technologies, due to the need of replacement, which is a challenge as the demand for bandwidth skyrockets due to video, mobile, and Internet of Things (IoT) applications. Service providers aim to expand and scale their networks while keeping costs in check. However, traditional devices impose limitations that hinder scalability, raise deployment costs, and reduce operational efficiency [6].

The industry is undergoing a shift from hardware-centric to virtualized, software-based networks. A critical facilitator of this transition is the NFV approach, which involves replacing dedicated physical network devices with software programs running on standard computer hardware to perform the same network functions. This evolution is driven by the need to address the previously mentioned requirements and constraints [6].

In Section 2.2, the introduction of NFV, its architecture and its advantages in addressing the aforementioned limitations are discussed.

2.2 Network Function Virtualization

NFV enables the replacement of physical network devices performing specific NFs with one or more software programs executing the same network functions while running on generic computer hardware. NFV decouples the software from hardware, offering the ability to use any commercially available hardware to implement the VNFs [6].

According to [3], [6], some of the NFV architecture advantages are:

- Hardware flexibility, since Network Operators have the freedom to choose and build the hardware, reducing equipment cost;
- Agility, by rapidly deploying, terminating, reconfiguring or changing the topological location of a VNF;
- Decrease of the time-to-market, minimizing the lifecycle of creating a network system;
- Scalability and Elasticity, due to the capability of a VNF to expand and stretch resources or to scale them down according to the requirements;
- Test-setups, without the need of creating a replica of the production environment to run in-house tests.

In Europe, the organization tasked with setting the standards for NFV is the ETSI. In 2012, they established an Industry Specification Group (ISG) dedicated to researching and providing guidelines for NFV [7].

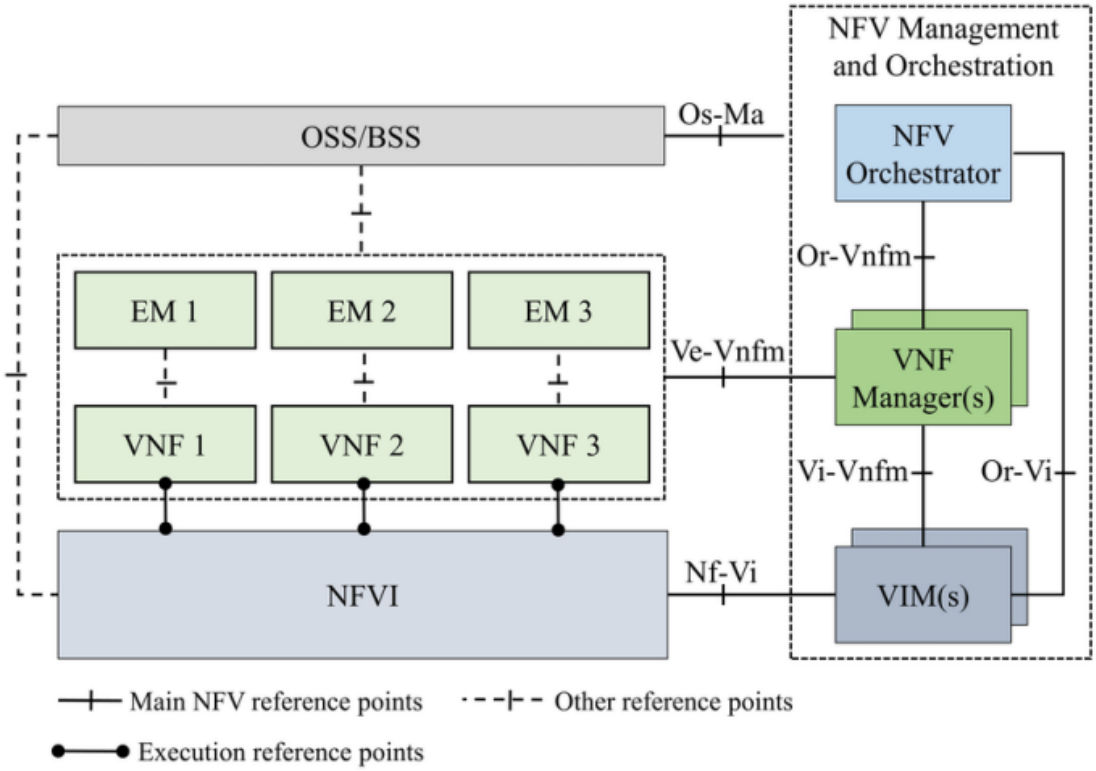

Figure 2.1: NFV Reference Architectural Framework [8]

The latest release is Release 4 which addresses the work developed in 2019 and 2020 with a set of new features, among them are [9]:

- NFV-MANO automation and autonomous networks;
- NFV enhancements for 5G;
- SBA for NFV-MANO;
- Continuous VNF integration.

2.2.1 ETSI MANO

MANO is one of the most relevant elements of the ETSI NFV architecture, being responsible for the orchestration and lifecycle management of the resources involved in NFV.

Figure 2.1 depicts the ETSI NFV architectural framework, which includes MANO.

This framework consists of three main functional blocks [6], [10]:

- **Network Function Virtualization Orchestrator (NFVO)** - responsible for the orchestration of Network Function Virtualization Infrastructure (NFVI) resources across multiple Virtualised Infrastructure Managers (VIMs) and the lifecycle management of Network Services.
- **VNF Manager** - in charge of the lifecycle management of VNF instances.
- **VIM** - responsible for controlling and managing the NFVI compute, storage, and network resources.

There are several NFV MANO implementations, developed by different projects like ONAP and OSM, which are going to be addressed next.

2.3 MANO Solutions

2.3.1 ONAP

ONAP is a open-source project founded by AT&T and China Mobile, now under the Linux Foundation. It was formed through the combination of the Enhanced Control, Orchestration, Management & Policy (ECOMP) and Open Orchestrator (Open-O) projects into ONAP [11]. It aims to provide a comprehensive platform for automating and managing networks, with a focus on Software-Defined Networking (SDN) and NFV.

ONAP allows organizations and their network or cloud providers to collaboratively instantiate network elements and services in a dynamic process with real-time responses to events. Some key components of the architecture are [12]:

- **Design Time Environment** - offers tools, techniques and repositories for defining resources, services, and products.
- **Runtime Environment** - provides a model- and policy-driven orchestration and control framework for an automated instantiation and configuration of services and resources.
- **Shared Services** - gives ONAP's modules shared capabilities and utilities.
- **External API** - provides northbound interoperability for the ONAP Platform.
- **OOM** - provides the ability to manage cloud-native installation and deployments to Kubernetes-managed cloud environments.

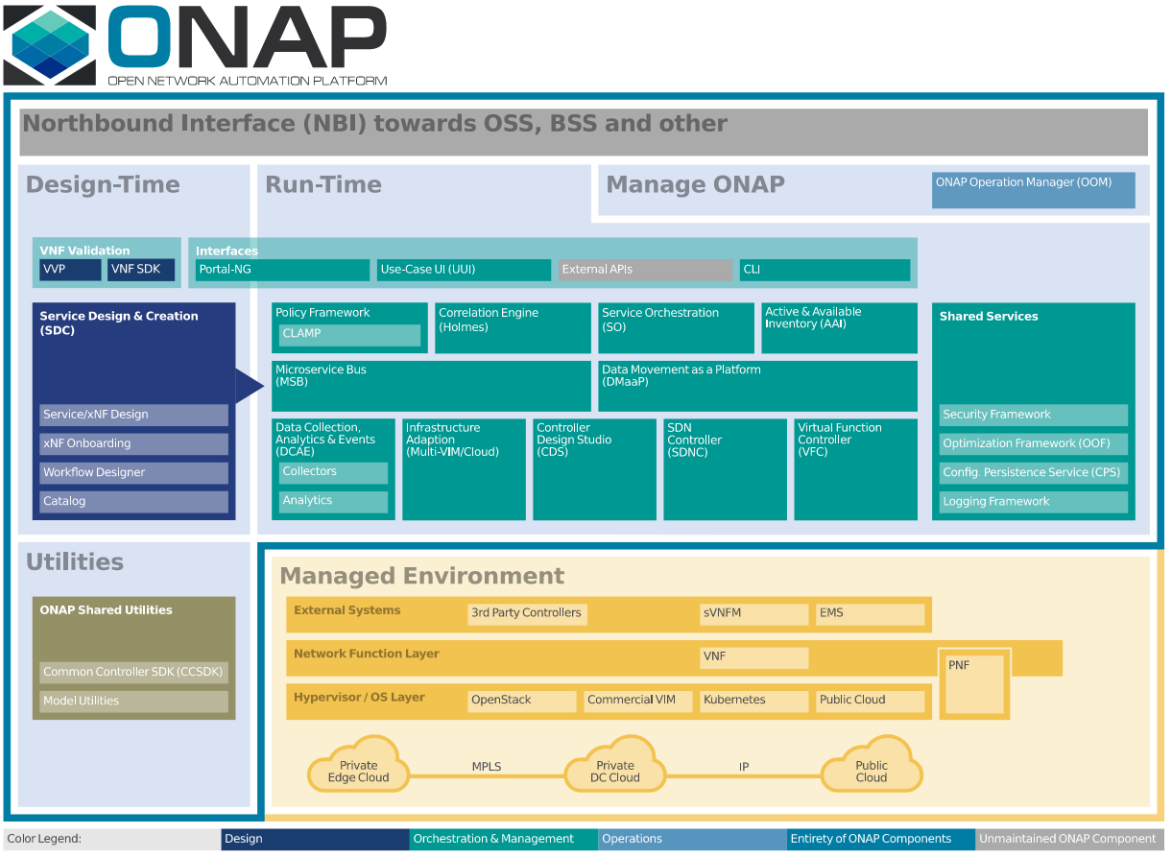

Figure 2.2: ONAP's London Release - Platform Architecture Diagram [12]

Figure 2.2 presents the architecture of the most recent ONAP's release, where the components of each framework mentioned before are displayed.

2.3.2 OSM

The OSM project, led by ETSI, aims to create an open-source solution for managing and orchestrating NFV in order to assist Telecommunication Operators in deploying NFV solutions efficiently and cost-effectively. As it is an ETSI initiative, OSM is fully compliant with ETSI's NFV standards [13].

Starting from Release 4, the OSM project has included the OSM Northbound API which is standardized in ETSI GS NFV-SOL 005 [14]. This API allows Operations Support Systems (OSS) and Business Support Systems (BSS) to interact with the NFVO through RESTful APIs. The OSM Release 9, which was released in December 2020, also included the support of a new standard, ETSI GS NFV-SOL 006 [15], which replaces the syntax used in VNF and Network Service (NS) descriptors [16], [17].

Regarding some interesting added features from the previous releases, worth of mentioning, Release 7 added Containerized Network Functions (CNFs). Since then, OSM can model the deployment of Kubernetes[1] workloads using Juju[2] bundles or Helm[3]. Until Release 8, only Helm version 2 was supported, changing in the next release, 9, to support Helm version 3. Release 10, brought the ability for on-demand horizontal scaling of Juju bundle-based Kubernetes workloads, enabling applications to scale-out to provide additional capacity and release their resources, or scale in, as usage returns to the baseline [17], [18].

Currently OSM is at Release 14, but the 5GASP project is adopting the Release 12.

The ETSI MANO main components were already addressed in Section 2.2.1, so the main approaches of OSM will be addressed next.

Figure 2.3 represents the OSM Main Architectural Components.

The OSM consists of two decoupled orchestrators namely, Resource Orchestrator (RO) and Network Service Orchestrator (NSO). The RO provides orchestration in the SDN and cloud technology domains. The NSO handles lifecycle management of network services and VNFs and consumes information models such as Yet Another Next Generation (YANG) [20].

The strategy of OSM focuses on reducing the complexity of integration through four main principles [21]:

- A well-known Information Model (IM) that enables the modeling and automation of the entire lifecycle of NFs, NSs, and Network Slices.
- A unified Northbound Interface (NBI), based on ETSI GS NFV-SOL 005, which enables the full operation of system and the NSs and Network Slices under its control.
- The extended concept of Network Service in OSM, allows for the integration of an NS across various domains, including virtual, physical, and transport. This extension enables control over the entire lifecycle of a NS while interacting with VNFs.

[1] https://kubernetes.io/
[2] https://juju.is/
[3] https://helm.sh/

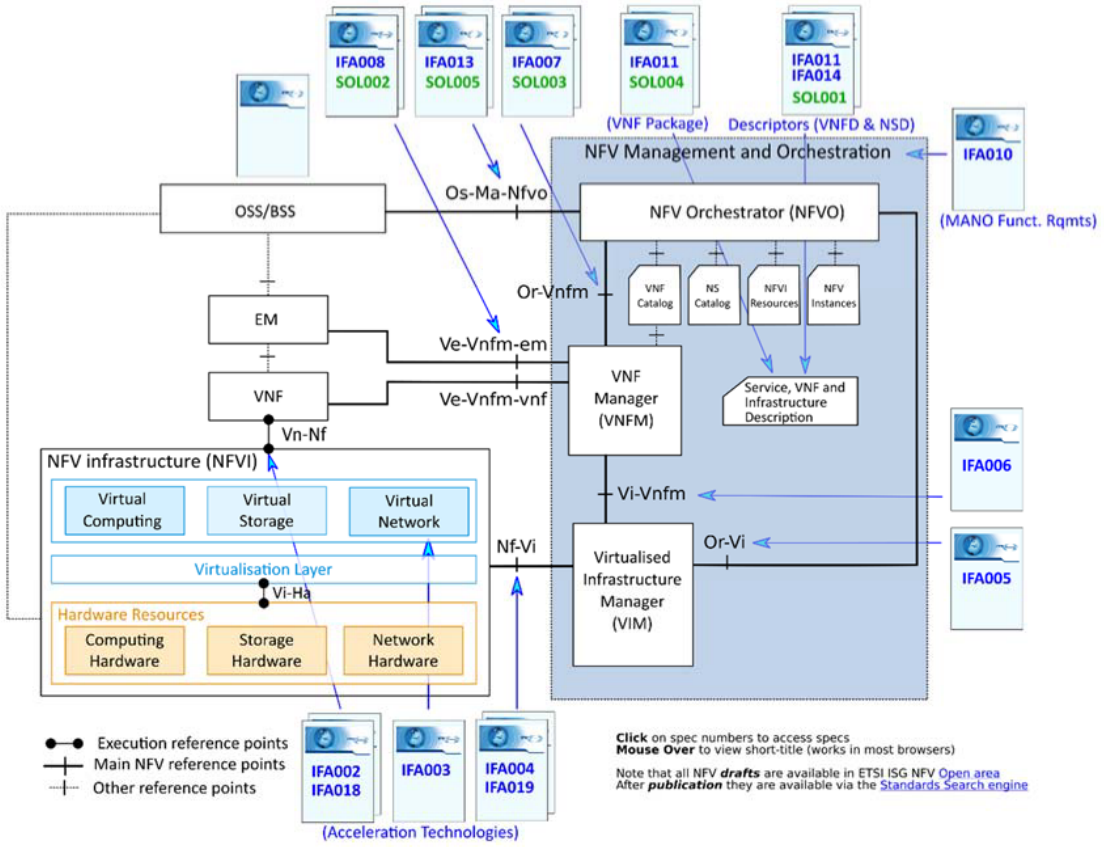

Figure 2.3: OSM Main Architectural Components [19]

- In addition, OSM can also manage the lifecycle of Network Slices, assuming if required the role of Slice Manager, extending it also to support an integrated operation.

2.4 Network Applications

The concept of Network Application emerged in the 5GPP Horizon 2020 ICT-41 calls, initially lacking a precise and comprehensive definition within the established standards and the broader 5G community. Consequently, there arose a need for a shared definition among the ICT-41 projects to establish a common understanding of this term.

As a result, it refers to a set of services, deployed over 5G infrastructures, that provide specific functionalities tailored to various industries and their associated use cases. Moreover, a Network Application is defined as a software component that interacts with the control plane of a mobile network by consuming standardized APIs from the 5GC, Radio Access Network (RAN) Intelligent Controller, and edge computing. These applications provide services to vertical industries either by integrating within their existing systems or by exposing business APIs [22], [23].

Additionally, the Network Application ecosystem aims to simplify the implementation and deployment of vertical systems on a large scale. It involves a middleware layer that allows third parties and network operators to contribute to the development of Network Applications

based on the level of interaction and trust. In certain cases, Network Applications residing in the operator domain can have additional access to network capabilities beyond what is available to third-party Network Applications [23].

Nonetheless, it is imperative to address the requirement of testing and validating Network Applications within a 5G virtualized experimental environment, encompassing diverse implemented functions and vertical-specific configurations. This crucial step guarantees the desirability and viability of these applications for Telecommunication Operators, thereby fostering their inclination to embrace and utilize them [24].

2.5 5G Core System Architecture

The 3GPP specification for the 5G System Architecture has been formulated to facilitate data connectivity and services, thereby allowing deployments to implement technologies like NFV and SDN. Some fundamental principles and concepts in this regard include [25]:

- Separate the User Plane (UP) and the Control Plane (CP) functions for scalability and flexible deployment.
- Modularize the function design for efficient network slicing.
- Define procedures, i.e. the set of interactions between NFs, as reusable services.
- Implement unified authentication.
- Support stateless NFs for resource flexibility.
- Enable capability exposure.
- Allow concurrent access to local and centralized services.

Moreover, in 5GC, a significant departure from traditional network architectures is the adoption of service-based interactions between NFs, replacing the traditional "nodes" and "network elements" connected by interfaces. Each NF in 5GC provides services to others, accessible via APIs over NF interfaces within the common SBA. This approach enables specific NF functionalities to be easily accessed and utilized by others [5].

The 3GPP TS 23.501 [26] specifies the overall 5G SBA for the core network, which involves the conversion of traditional network elements into NFs. Figure 2.4 depicts this architecture in release 16.

The control plane is composed by NFs that offer their services to any other applicable NFs via a common framework of interfaces accessible to all [27].

Figure 2.4 includes the following NFs and other modules [27] [28]:

- **Authentication Server Function (AUSF)** - Performs authentication between User Equipment (UE) and the network.
- **Session Management Function (SMF)** - Responsible for Protocol Data Unit (PDU) session establishment, modification, and release between a UE and a data network. Also interacts with the charging and policy systems.
- **Access and Mobility Management Function (AMF)** - Receives all connection and session-related information from the UE, but is responsible only for handling connection, registration, reachability, and mobility management tasks. All messages related to session management are forwarded to the SMF.

- **Policy Control Function (PCF)** - Provides functionalities for the control and management of policy rules, including rules for Quality of Service (QoS) enforcement, charging and traffic routing.
- **Network Repository Function (NRF)** - It helps to provide the service discovery function. Allows NFs to register their functionality and to discover the services offered by other NFs present in the network.
- **Network Slice Selection Function (NSSF)** - Allows service providers to supply specific portions of their network for specific use-cases, i.e., selects the set of network slice instances to accommodate the service request from a UE.
- **Network Slice Specific Authentication and Authorization Function (NSSAAF)** - Performs authentication and authorization specific to a slice.
- **NEF** - Provides an interface for outside applications to communicate with the 5G network to obtain network-related information about monitoring, provisioning, analytics reporting and policy and charging.
- **Unified Data Management (UDM)** - Responsible for access authorization and subscription management.
- **User Plane Function (UPF)** - Handles the user plane path of PDU sessions.
- **Application Function (AF)** - Provides session-related information to the PCF so that the SMF can ultimately use this information for session management.
- **UE** - Allows a user access to network services.
- **(Radio) Access Network ((R)AN)** - Provides access to a 5G core network. This includes the 5G RAN and other wireless and wired access networks.
- **Data Network (DN)** - Allows UE to be logically connected by a session.
- **Service Communication Proxy (SCP)** - The SCP enables multiple NFs to communicate with each other and with user plane entities in a highly distributed multi-access edge compute cloud environment. This provides routing control, resiliency, and observability to the core network.

The goal of this model is to maximize the modularity, reusability, and self-containment of NFs, and to enable flexible growth through the use of NFV and SDN. This allows for greater flexibility and scalability in the network [27].

3GPP has selected the Representational State Transfer (REST) architectural design model to support the design and communication between the distributed components the 5GC [28].

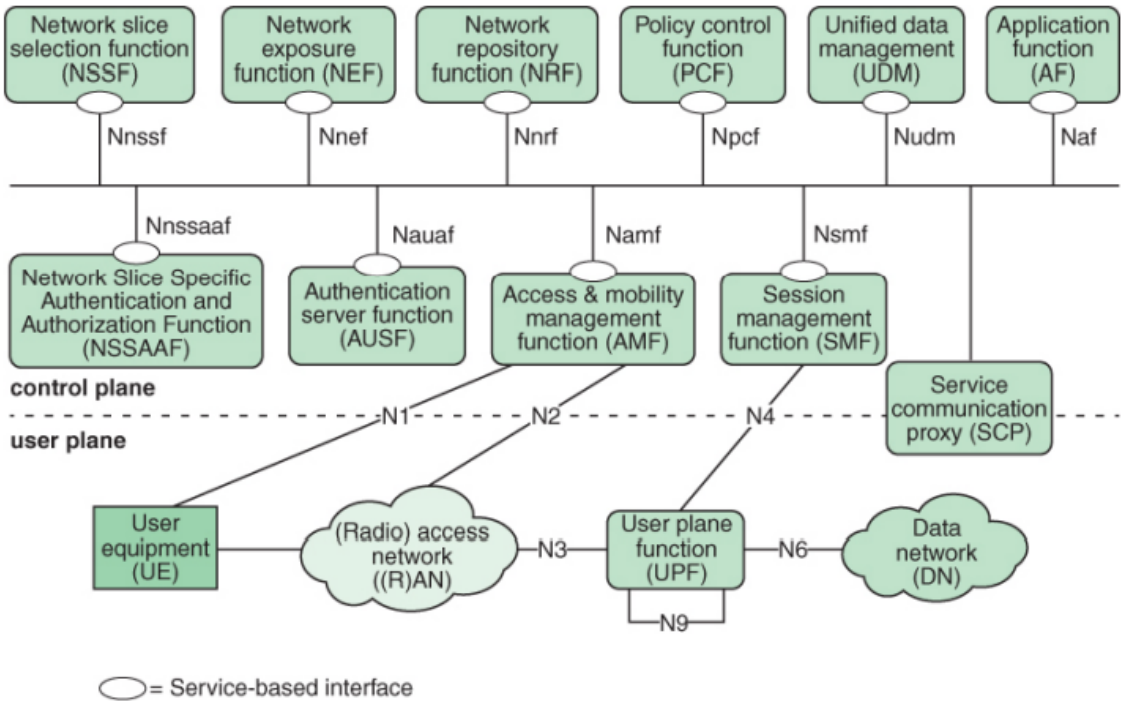

Figure 2.4: 5G Core System Architecture [27]

2.5.1 5G Core Network Capability & Network Exposure Function

One of the key NFs of the 5GC is the NEF, which provides secure means for attaching services and exposing capabilities provided by other 5GC NFs to external systems. This brings the concept of openness by enabling new levels of programmability in the core and edge domains, which allows third-party innovators, also known as vertical industries, to control and monitor the deployment process and performance of their services [29].

The exposure of the 5G capabilities is securely made through the NEF's APIs, hiding all the underlying network topology, to external third parties that may use them to build network aware applications. Furthermore, the internal communication between the NFs, that compose the SBA, is also made via the Northbound APIs [30].

To delve into further details, the NEF comprises several services described as Monitoring services, Policy and Charging Services, Application Provisioning Services, Analytics Services and IoT specific services

This services can be categorized as follows [30]:

- **Monitoring Services** - Provides monitoring capabilities for UEs such as loss of connectivity, reachability, location, access type, roaming status, etc.
- **Policy and Charging Services** - Enables to request QoS policies to utilize the radio resources efficiently and provides information about the network status.
- **Application Provisioning Services** - Provides applications with the means to assist the network by providing application related information that may provoke the 5GC to take different actions.

- **Analytics Services** - Allows efficient QoS management, traffic steering and mobility management by exposing network analytics.
- **IoT Services** - NEF exposes APIs that allow the external application to exchange data with the UE using different non-Internet Protocol (IP) data delivery methods: Short Messaging Service (SMS) and Non-Access Stratum (NAS) based Non-IP Data Delivery (NIDD).

2.6 Agile Methodology

2.6.1 DevOps

DevOps, introduced by Patrick Debois, Gene Kim, and John Willis in 2007-2009, is a set of practices that blend Development and Operations, fostering team collaboration. It reduces barriers between developers, who aim to innovate and deliver quickly, and operations, focused on ensuring production system stability and the quality of system changes, all while enhancing business value and competitiveness through faster user-oriented results. This approach extends the principles of agile processes [31], [32].

To enhance collaboration and communication between Development and Operations, it's vital to incorporate key elements which include [31], [32]:

- Increased frequency of application deployments through CI/CD.
- Automated unit and integration testing.
- Establishment of user feedback collection mechanisms.
- Ongoing monitoring of applications and infrastructure.

As illustrated in Fig. 2.5, to attain the aforementioned elements, DevOps relies on three fundamental pillars as its core foundation [31]:

1. **Collaborative Culture** - At the heart of DevOps is the dissolution of specialized teams like developers, Ops, and testers that worked separately. Instead, multidisciplinary teams with a shared objective come together to swiftly deliver added value to the product
2. **Processes** - Achieving swift deployment requires the implementation of agile methodologies, featuring iterative phases for enhanced functionality, quality, and prompt feedback. The DevOps process comprises several key stages such as planning, development, continuous integration, delivery, deployment, and monitoring.
3. **Tools** - The selection of tools and products holds immense significance. The conventional approach of segregated teams, each relying on their specialized tools, has given way to a more integrated and cooperative toolset, bridging communication gaps.

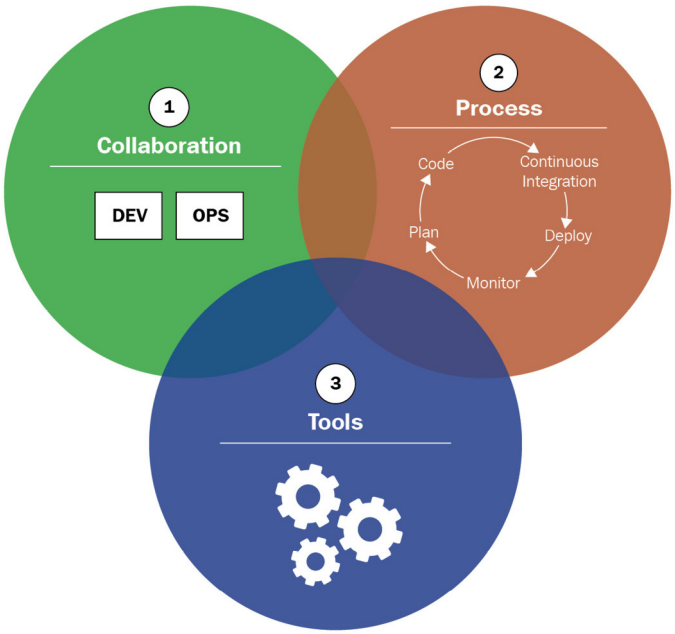

Figure 2.5: DevOps Axes [31]

2.6.2 CI/CD

CI is the critical initial step for ensuring consistent and high-quality software. It involves maintaining the software in a deployable state, including rigorous checks to ensure that the code compiles and maintains a reasonable level of quality at all times. This approach validates software immediately upon check-in to source control, offering assurance that it functions correctly and remains reliable even as new code is added [33].

CI relies on storing application code in a central repository that keeps track of all the changes committed by the developers. Each code change initiates an automated build process, generating a consistent build artifact stored in a core repository. This reproducible build process guarantees the same outcome with every execution from the same code [34].

CD builds upon CI by maintaining a consistently stable code mainline, allowing production deployment at any time. This reliability is upheld through automated deployment and testing, ensuring a single artifact is built and stored centrally. Testing and deploying across different environments follow the same standardized process, using the identical artifact. Automated testing on a production-like test machine ensures consistency, backed by comprehensive tests to meet both functional and nonfunctional requirements [34].

In conclusion, through the implementation of CI and CD, the software supply chain is automated, allowing for the rapid delivery of application code to production. This, in turn, yields secure and higher-quality code, delivers faster feedback, and ultimately shortens the time-to-market for the product [34]. Fig. 2.6 portrays the CI/CD pipeline.

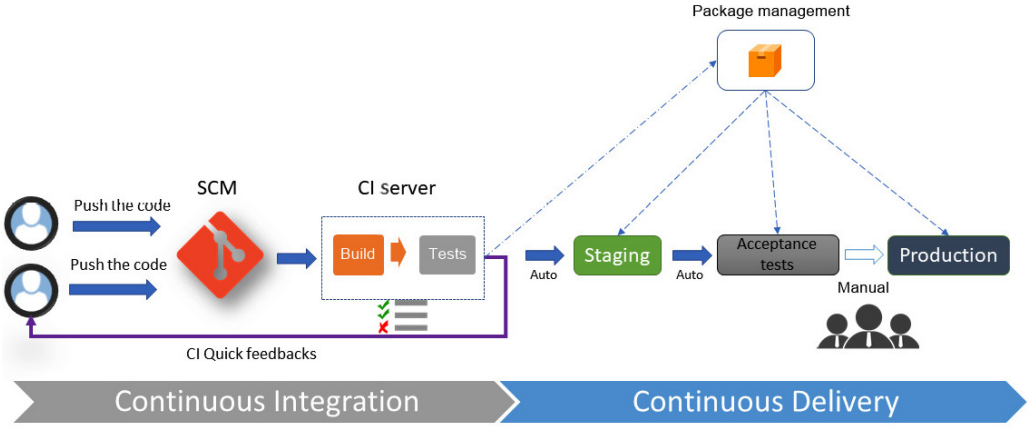

Figure 2.6: CI/CD Pipeline [31]

2.6.3 Harnessing DevOps and CI within a NFVContext

As refered in Section 2.1, organizations need to adapt their strategies to align with demands for speed, automation, flexibility, rapid time-to-market, and cost-efficiency, due to the requirements of 5G.

The deployment and configuration of a complete NFV infrastructure for obtaining its full benefits is a complex and time-consuming endeavor. To simplify NFV operations, automation is crucial, reducing human intervention, eliminating barriers between VNF providers and service providers, and enabling incremental VNF deployment [35], [36].

DevOps plays a crucial role in this context. It represents a modern software engineering culture characterized by automation, collaboration, and communication between software developers and network infrastructure operators. DevOps offers a well-defined set of methodologies, tools, and platforms for programming and service execution [35], [36].

When integrated with NFV service providers, DevOps reduces the manual effort required for onboarding VNFs and configuring infrastructure by automating processes. As a result, it accelerates the time-to-market for new services [35], [36].

Combining DevOps with NFV unlocks the potential to [35]:

- Automate the deployment and testing of newly updated network services.
- Perform testing and benchmarking on components within the NFV infrastructure.
- Achieve efficient VNFs onboarding, deployment, scaling, and resilience through policy-driven, rapid processes.
- Automatically fine-tune VNFs deployment parameters, including Central Processing Unit (CPU) and Random Access Memory (RAM), during deployment or runtime.
- Enforce real-time monitoring of anticipated VNFs operational behavior within the NFV infrastructure.
- Automatically adjust resource scaling and apply patches based on real-time monitoring events.

The exploration of this topic can be further delved into by referring to ETSI GR NFV-TST 006 [37], which outlines the ETSI perspective on integrating CI/CD and DevOps methodologies to validate VNFs.

2.7 Testing Automation Frameworks

Automation of the testing and validation processes is needed to minimize human intervention as much as possible. Several automation tools exist to validate the behavior of an application and provide results that application developers can consult. Their aim is to increase the application's test coverage, ensuring that the developed tests are reusable, easy to maintain, and stable. Relying on Testing Automation Frameworks allows for the automatic execution of code analysis, unit tests, integration tests, and more [38].

These frameworks enable improved testing efficiency and accuracy, resulting in reduced testing application costs and offering the following benefits: (i) decreased human intervention, (ii) code reusability in testing, (iii) reduced maintenance and test development expenses, and (iv) enhanced test efficiency [38].

Currently, several frameworks are used, such as Selenium[4], Cypress[5], Appium[6], Robot Framework[7], and WebDriverIO[8], among others.

The Robot Framework will be described in more detail in this work, as it was the tool used to develop the tests explained later in this document. This choice was made because the tool was already in use in the 5GASP project.

The Robot Framework is an open-source, versatile, and user-friendly test automation framework designed to automate test cases and processes. It has an easy syntax, utilizing human-readable keywords [39].

One of the notable strengths of the Robot Framework is its extensibility. Users can create custom libraries and keywords using Python or Java, allowing for the adaptation of a wide range of systems and applications. It also includes built-in libraries for common tasks, reducing the need for extensive coding in many cases [39].

Concerning the testing process output, Robot Framework produces Extensible Markup Language (XML) and HyperText Markup Language (HTML) interactive report files.

2.8 Related Work

The European 5G Infrastructure Public Private Partnership (5GPPP) has several projects focused on creating end-to-end 5G platforms for automated validation of Network Applications and services for 5G developers and experimenters. In this Section it will be highlighted recently completed or ongoing projects that aim to achieve this goal, such as (i) VITAL-5G[9], (ii)

[4] https://www.selenium.dev/
[5] https://www.cypress.io/
[6] http://appium.io/docs/en/2.1/
[7] https://robotframework.org/
[8] https://webdriver.io/
[9] https://www.vital5g.eu/

EVOLVED-5G[10], (iii) 5GMediaHUB[11] and lastly the project that this dissertation is part of, (iv) 5GASP.

2.8.1 VITAL-5G

The VITAL-5G project, which is part of the H2020 program (ICT-41-2020), runs alongside 5GASP and focuses on enhancing Transportation and Logistics (T&L) services. It involves Small and Medium-sized Enterprisess (SMEs) and stakeholders in a collaborative effort to provide an open, secure, and virtualized 5G environment enabling testing, validation, and verification of Network Applications [40].

The project's main goal is to establish a flexible experimentation facility with an intelligent virtual platform and three distributed European 5G-testbeds with T&L facilities that cover scenarios spanning from river and sea-ports to warehouses. These resources, along with associated vertical infrastructure, allow real-life testing of T&L Network Applications using 5G connectivity [41].

To validate 5G services provides a Testing & Validation Framework and a detailed procedure to do so that are explained throughout this Section. Also, the use cases are described in this Section. In the context of this dissertation, VITAL-5G defined a methodology for defining metrics that will be address.

Testing & Validation Framework

The VITAL-5G Testing & Validation Framework, illustrated in Fig. 2.7, provides advanced testing capabilities using specialized monitoring and testing tools specific to the testbed, along with globally accessible platform testing tools. This allows for the validation of network and vertical services across the three VITAL-5G trial sites. Experimenters can access these capabilities centrally through the VITAL-5G Portal. They can define their experiments, set up specific test cases, configure validation parameters, and receive validation reports through the portal [42].

The most crucial element of the entire process is the Testcase template, enabling experimenters to specify test details, configure network and infrastructure elements, define metrics and Key Performance Indicators (KPIs), and outline validation conditions for the experiment [42].

Two additional modules play a significant role in this process [42], [43]:

- **Results Analytics (RA)** - This module gathers experiment-generated metrics, computes chosen KPIs designated by the experimenter, and verifies them against predefined acceptable parameters.
- **Performance Diagnosis (PD)** - The PD module gathers, parses, and analyzes experiment data, providing insights into the performance of different experiment components. It identifies anomalies, traces the source of issues, and diagnoses potential problems or performance reductions encountered during the experiment.

[10] https://evolved-5g.eu/
[11] https://www.5gmediahub.eu/

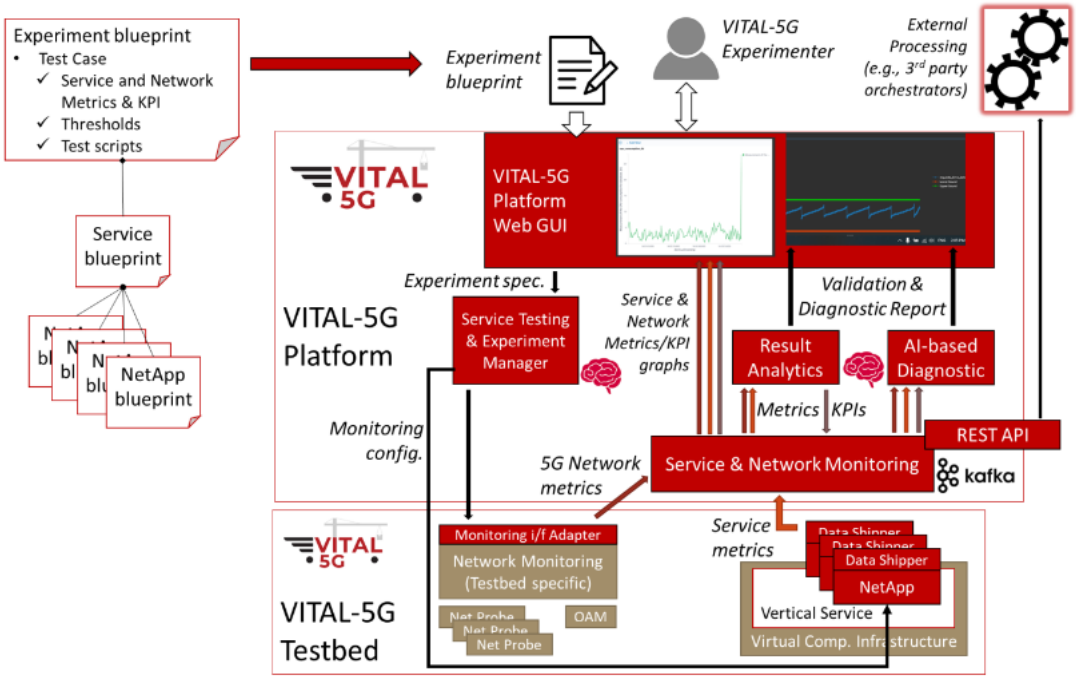

Figure 2.7: VITAL-5G Testing & Validation Framework [42]

Testing & Validation Procedure

The testing and validation process within the experimentation service involves defining, executing, and monitoring tests. It includes collecting and processing metrics, ensuring successful test execution, and evaluating outcomes against experimenter-defined validation KPIs to validate the service under test [42].

Pictured in Fig. 2.8, the VITAL-5G Testing & Validation process is divided into four distinct stages [42]:

1. **Experiment Design** - This phase involves defining the experiments in detail, whether they relate to validate a specific T&L service or evaluate network performance under specific circumstances. The Testcase Template is introduced here, providing the necessary context and information for the specific tests.
2. **Experiment Execution** - In this stage, a range of local and global monitoring tools are activated to continuously gather data in line with the measurements, requisites, and KPIs stipulated in the Testcase by the experimenter. Real-time visualization of the experiment's progress is facilitated during this phase.
3. **Results Evaluation & Service Validation** - The network and/or service components' performance undergoes evaluation, and service functionalities are validated in alignment with the predetermined KPIs and the evaluation and validation criteria set forth by the experimenter. During this phase, the RA module and the PD module are activated. Their purpose is to analyze the gathered results, offering valuable perspectives on the performance of the chosen components and potentially the overall end-to-end service.

4. **Experiment Finalization** - This phase concludes the experimentation activities, ensuring proper closure, and features result visualization and reporting.

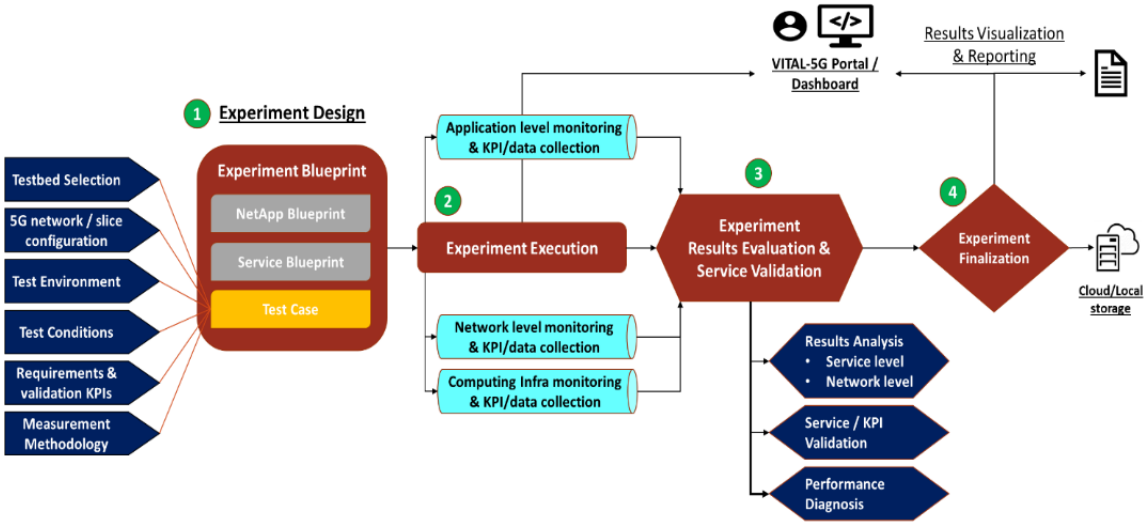

Figure 2.8: VITAL-5G Testing & Validation Procedure [42]

European Testbeds & Use Cases

VITAL-5G focuses on vertical organizations within the T&L sectors. It provides three testbeds containing T&L facilities that focuses in scenarios ranging from river and sea ports to warehouses. Each facility is enhanced with pre-commercial 5G network installations. These networks supply the necessary mobile connectivity for T&L services and incorporate advanced features like network programmability, slicing, and monitoring. Additionally, they are coupled with virtual computing resources to execute service applications. This approach enables the controlled and adaptable replication of various operational conditions. As a result, vertical organizations can design and execute diverse experiments to validate the functionalities and performance of their T&L services in specialized and adjustable 5G environments [44].

The project revolves around three distinct use cases [41], [44]–[46]:

- **Automated Vessel Transport** - This use case focuses on demonstrating the automation of logistics transport via vessels, utilizing advanced 5G communication technology. The Port of Antwerp serves as the execution site, where 5G connectivity and slicing are leveraged to control semi-autonomous vessels within the challenging port environment. Real-time transmission of high-bandwidth camera feeds and sensor data occurs from the vessels to a central command center, which then sends real-time steering commands back to the vessels.
- **5G Connectivity and Data-Enabled Assisted Navigation Using IoT Sensing and Video Cameras** - This use case involves the implementation of an assisted navigation system using IoT sensing and video cameras. The system is installed at a river port, the Galati port, as well as on ships and barges. The objective is to enhance

safety and security during river port operations and ship navigation, even in adverse weather and water conditions.

- **Automation and Remote Operation of Freight Logistics** - Based in a Greek depot, this use case covers all standard warehouse operations, including receiving, stocking, picking, checking, packaging, labeling, and shipping services. The aim is to showcase the applicability of 5G technology within a comprehensive logistics context. The integration of Automated Guided Vehicles (AGVs) in this operational system aims to optimize warehousing processes.

Overall, these use cases collectively showcase the potential of 5G technology to revolutionize various aspects of transportation and logistics, from vessel control and navigation to warehouse operations.

KPIs

The project's extensive range of pilots and the inclusion of diverse 5G ecosystem elements, such as the 5G radio, core network, NFV infrastructure, VITAL-5G platform, and services for each use case, prompted a analysis of essential metrics. These metrics are pivotal for evaluating and validating the performance of 5G-boosted vertical services in port and warehouse contexts [47].

To achieve this, four distinct metric categories were established, each aligned with the broader scope of the 5G ecosystem, they are related to: network, infrastructure, platform, and service metrics [47].

A standardized methodology was created to define metrics across all categories, including metric type, measurability, priority, description, testbed components, stakeholders, measurement frequency, source, and specifics. Non-Functional metrics assess vertical service deployment and management efficiency, unrelated to end user experience, e.g., vessels or AGVs. Functional metrics assess network, infrastructure, platform, and service performance, directly impacting user perception. Measurability and priority are evaluated for each metric to determine complexity and importance in delivering 5G-powered vertical services in VITAL-5G. Involved testbed components pinpoint measurement elements, while parties responsible for metrics are tracked. Measurement granularity reflects real-time or non-real-time assessment, often in seconds, minutes, or days. Measurement description are detailed in test cases, outlining procedures for tests and data collection [47].

Fig. 2.9 shows how Vital-5G defines metrics.

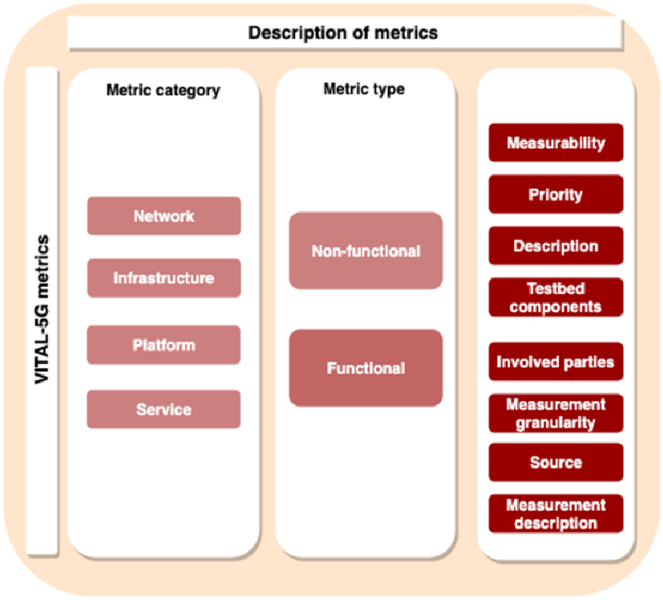

Figure 2.9: Methodology of defining metrics in the Vital-5G [47]

2.8.2 EVOLVED-5G

The EU-funded EVOLVED-5G project, under the ICT-41-2020 call, aims to upgrade the 5G experimentation potential in Europe and materialise the openness of 5G to vertical industries by creating a Network Application development and verification environment[48]. This project will mainly contribute towards the fourth industrial revolution or Industry 4.0, allowing for the development of industry-oriented Network Applications which will be tested, validated and certified in a vendor-agnostic experimentation platform. It builds upon the concept that SBA allows the exposure of network services and capabilities through the NEF, a border function of the 5GC network, that enable Network Applications to consume APIs that combined compose a new service [49].

The components of the EVOLVED-5G facility are structured into clusters referred to as environments, each corresponding to a specific stage in the lifecycle of the Network Application. These stages include development and verification (Workspace Environment), validation (Validation Environment), certification (Certification Environment), and release to the Marketplace, which is also treated as an environment. Moreover, the 5G Non-Public Networks (NPN) Environment plays a crucial role in facilitating both the validation and certification stages within the Network Application's lifecycle [50].

Figure 2.10 illustrates the architecture of the EVOLVED-5G system, followed by an elaboration on the included environments [50], [51].

- **Workspace Environment** - This environment functions as a collaborative space for Network Application implementation, equipped with tools and functionalities to facilitate development and verification for functional testing of Network Applications. It involves two distinct phases in the Network Application lifecycle: Development and

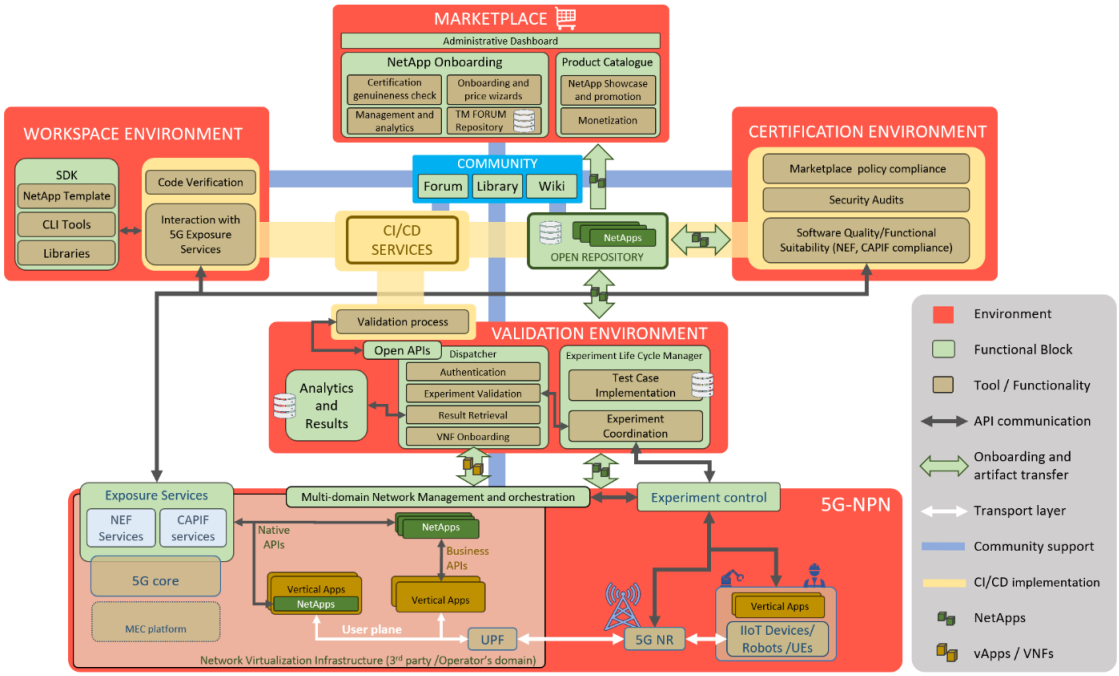

Figure 2.10: EVOLVED-5G System Architecture [50]

Verification.

- **Validation Environment** - This environment focuses on enabling the connection between the Network Application and the Vertical Application through controlled scenarios that simulate real operating conditions for the Network Application. Referred to as the Validation phase in the Network Application lifecycle, this stage involves a series of coordinated tests agreed upon with the Vertical Application provider. The objective is to ensure the correct functioning of both the Network Application and the Vertical Application in a more realistic setting that utilizes an actual 5G infrastructure. Notably, the validation process covers the entire service provisioning chain, evaluating not only Network Application functionality but also assessing the infrastructure as a whole, including the extraction of performance KPIs.
- **Certification Environment** - In contemporary virtualized network domains, the significance of conformance tests and quality evaluations cannot be overlooked. This environment is explicitly tailored to fulfill this purpose. It comprises functional blocks that contribute to transforming a validated Network Application into a certified one, ensuring its compatibility with a 5G Stand Alone (SA) Network.
- **Marketplace Environment** - Within this environment, the final stage of the Network Application lifecycle occurs: Certified Network Applications are released to the market, becoming accessible to end users.
- **5G-NPN environment** - This environment's primary goal is to equip Network Applications with essential 5G native or northbound APIs. It also provides the necessary connectivity infrastructure to support the objectives of the Validation and Certification environments. It comprises both hardware and software components essential for

executing the validation and certification stages. These components involve ensuring Network Application interoperability with the 5G network, devices, and management and orchestration functionalities, among others. This layer, provides NEF and Common API Framework (CAPIF) services.

Interconnecting these five environments is made possible through (i) CI/CD services, (ii) a central repository referred to as the Open Repository, which includes repositories for both source code and artifacts, and (iii) an added element that hosts services promoting community development within the EVOLVED-5G ecosystem [50]. This configuration is depicted in Figure 2.11.

Network Application Lifecycle

The relation between the architectural components and the different phases of the Network Application lifecycle is depicted in Fig. 2.11.

Subsequently, a brief description of each phase is provided [50], [51]:

- **Development phase** - Development of the Network Application takes place during this step.
- **Verification phase** - During this stage, the application's compatibility with the 5GC APIs is verified through testing.
- **Validation phase** - In this phase, both the Vertical Application and Network Application are jointly deployed within an authentic 5G infrastructure, and their combined functionality is subject to testing.
- **Certification phase** - This phase englobes the execution of certification audits.
- **Publication phase** - Certified Network Applications are readied for sale to end-users through publication and management in the Marketplace.

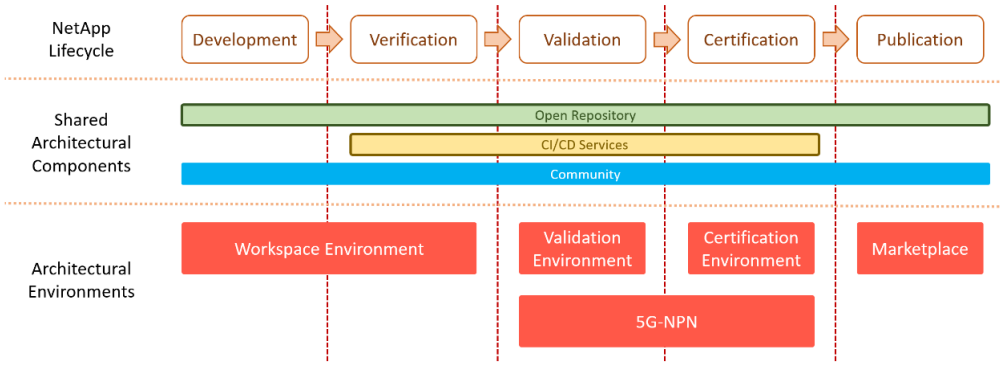

Figure 2.11: Network Application lifecycle phases and relation with architectural components [50]

Verification, Validation & Certification Phases

The toolkit employed to guarantee Network Application alignment with the EVOLVED-5G concept and framework involves a three-tier testing procedure, situated at the core of the

system through the Workspace (via the verification process), the Validation Environment, and the Certification Environment [50].

During the verification phase, initiated and monitored by the developer through the Command-Line Interface (CLI) Tool, interaction with the CI/CD Services component takes place. This component, functioning as a host and overseer facilitated by an orchestrator, manages essential verification tests crucial for evaluating Network Application functionalities, which include building, deploying, and destroying pipelines for Network Application containers [50].

Furthermore, the CI/CD Services component conducts other vital tasks, including static code analysis, vulnerability scans, connectivity assessments with NEF and CAPIF services , and validation of the anticipated communication ports for the Network Application, particularly with Vertical Applications and other services [50].

Within the verification phase, a diverse range of verification tests are executed by the CI/CD Services component. This includes assessments of code correctness, containerization success, 5G connectivity, and vulnerability scans. To facilitate this process, the CI/CD component handles the construction and deployment of the Network Application, as well as the NEF Emulator and CAPIF tool [50].

Upon completion of the verification process, the CI/CD Services component generates a results report. This report is then communicated back to the developer and presented through the CLI for review and consideration [50].

The validation phase follows the verification phase, with series of testing steps that build upon the verification process. It includes all the tests conducted during verification, serving multiple purposes. Firstly, it provides an evaluation of the Network Application, allowing developers to iteratively enhance its quality before moving to validation. Secondly, the validation phase tests the Network Application in a real environment, offering distinct insights compared to the verification phase. Thirdly, it permits auditing of results by platform operators, enabling early issue detection. Lastly, the validation ensures Network Application's operation under genuine or near-real network conditions, particularly with at least one Vertical Application [50].

The Validation Environment's frontend consists of the Dispatcher, an OpenAPI[12] implementation that manages authentication, experiment execution, and results retrieval. It provides a centralized entry point that handles user management and redirects requests to the appropriate backend component. Overseeing experiment execution, the Experiment Life-Cycle Manage (ELCM) defines the structure of test cases, outlining tasks and the sequence of actions required for conducting experiments. Results are stored in the Results Catalogue and can be analyzed using the Analytics Module, a web interface offering statistical analysis tools [50].

The validation process tests both Network Application functionality and performance in real network conditions, including Network Application-Vertical Application integration. Key aspects assessed during validation include [50]:

1. Successful completion of all verification phase tests.

[12]https://www.openapis.org/

2. Testing the Network Application within a real infrastructure and under actual or near-real network conditions in a 5G NPN.
3. Testing Network Application functionality for integration with a Vertical Application, whether it's specific to one Vertical Application or serves multiple.
4. Evaluating Network Application performance based on agreed-upon KPIs among Network Application developers, platform owners, and Vertical Application providers.

Finally, overseen by the Certification Authority via an accredited Certification lab, the certification phase is initiated by the entity seeking certification, usually the Network Application developers [50].

The proposed certification criteria are categorized into the following pillars [50]:

- **Software Product Quality** - Revolves around ensuring Network Applications' compliance with 3GPP standards for seamless integration and compatibility with 5G SA mobile core networks.
- **Security** - The certification goals related to security are centered on identifying vulnerabilities within the source code and deployed artifacts of Network Applications. Detecting vulnerabilities is essential to prevent breaches, fraud, maintain data privacy, and isolate data from the execution environment. Activities such as code auditing and application security analysis play a vital role in securing data exchange between Network Applications and the NEF core function, safeguarding sensitive user information.
- **Marketplace** - The Marketplace marks the ultimate stage in the commercialization of Network Applications, serving as the platform for publishing certified Network Applications. Similar to public cloud providers, these Marketplaces have specific policies that must be met before the Network Applications are published.

The certification execution environment within the EVOLVED-5G is treated as separate from the Validation Environment. Although they can share infrastructure and utilities, they operate with distinct configurations and provisioning setups [50].

While the validation process is thorough and accompanies minor updates to Network Applications, the certification process occurs infrequently and is prompted solely by specific recommendations or mandates. This infrequent execution approach helps keep costs low. Certification activities could take place periodically to enhance reinforcement or coincide with major software releases [50].

CAPIF Services

As the CAPIF Tool is one of the outcomes of the EVOLVED-5G project and it's part of the validation and certification of Network Applications, a brief description will be done in the following paragraphs.

The standardization of CAPIF ensures a singular reference model for provisioning services through APIs in 3GPP systems. The potential of this framework is due to the need for interaction between vertical industries and mobile networks [52].

More in detail, the CAPIF Tool will is a software component that emulates the CAPIF APIs of Release. 17 of TS 23.222 [53]. At its core, it implements REST APIs in order to

provide CAPIF services to Network Applications. CAPIF Tool enables Network Applications to consume CAPIF services to register Network Applications as API Invokers and consume 5G System APIs exposed, such as those exposed by the NEF APIs in the Emulator [51].

NEF-Emulator

In the context of the EVOLVED-5G ecosystem, NEF services hold crucial significance for the 5G NPN. The primary component of the project, Network Application, heavily relies on NEF services to expose business APIs to vertical applications. Hence, it's imperative that Network Applications can communicate seamlessly with the NEF services offered by the 5GC network. In order to expose the network's capabilities, NEF needs to interface with various NFs within the 5GC, referred to as Southbound APIs [52].

Currently, establishing communication with the southbound interface presents a significant hurdle, given that both the Athens and Malaga Platforms do not fully implement the SBA or the southbound interfaces required by NEF to provide standardized APIs. Nevertheless, the Emulator conquers this obstacle by generating simulated events. To elaborate, the NEF Emulator can be segmented into three distinct parts, as visualized in Fig.2.12. A comprehensive description of the primary characteristics of these components is described below [30], [52], [54]:

- **Exposure Layer** - It focuses on providing NEF APIs for 5GC functions, i.e, Northbound APIs, including monitoring events and session establishment with QoS. Additional APIs can be added later to this layer.
- **Simulation Environment** - Offers a geolocated environment for simulating 5G network scenarios, allowing users to test service APIs. This includes simulating UE movement for location monitoring, offering flexibility for Network Application developers to customize scenarios.
- **Common management Layer** - This layer includes token-based user authentication/authorization using OAuth2.0. Users can create accounts, and each Network Application developer has an isolated environment for configuration.

Moreover, the EVOLVED-5G NEF Emulator provides NEF services implemented as RESTful APIs, the ones that were implemented are defined in TS29.122 [55] and are [30], [50], [56]:

- **MonitoringEvent API** - This API allows a Network Application to access several events which may occur in the 5GC. Some preeminent examples include location reporting, loss of connectivity, and UE reachability. For instance, in the event of a UE handover to a neighboring cell, NEF notifies the Network Application about this occurrence, providing it with information about the new cell identifier to which the UE has migrated.
- **AsSessionWithQoS API** - The AsSessionWithQoS API will allow a Network Application to choose a predefined QoS profile from a list retrieved from the 5GC. Moreover,

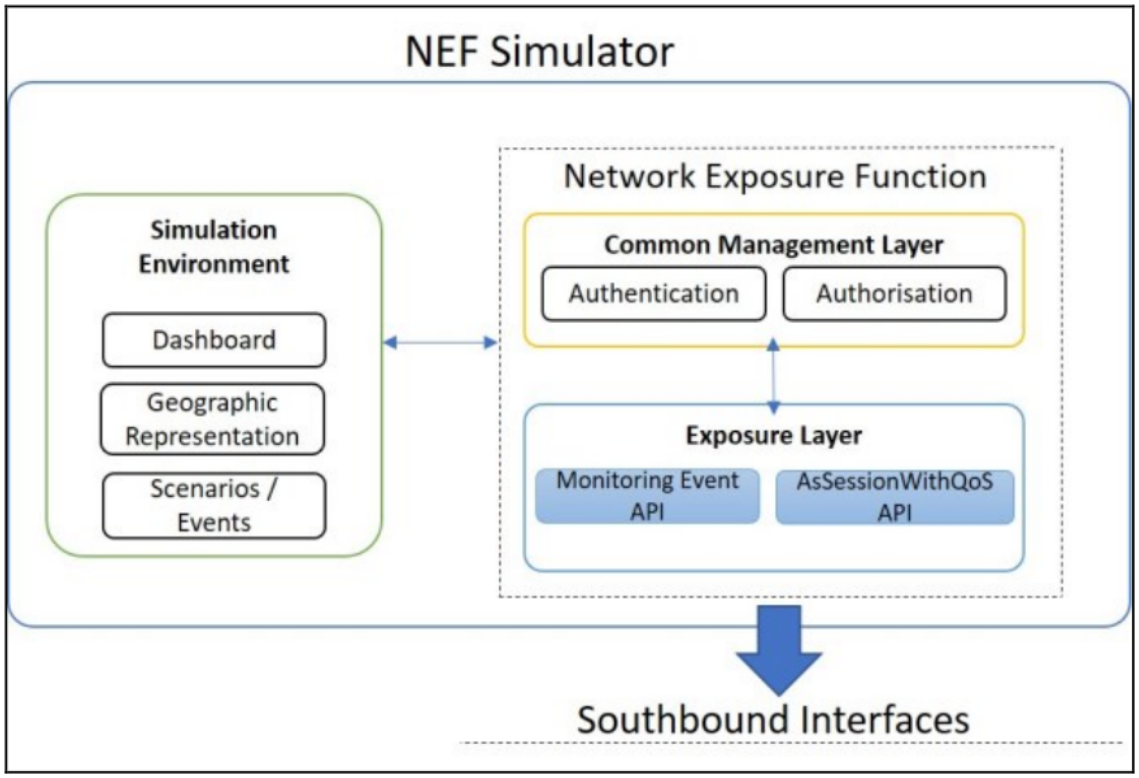

Figure 2.12: EVOLVED-5G NEF Emulator Architecture [30]

a Network Application can indicate the desired level of QoS, e.g., jitter, latency, and priority, for a given IP traffic flow.

The NEF Emulator is implemented using FastAPI[13], a modern, high performance web framework for building APIs with Python 3.6+. It is fully compatible with the open API standard, OpenAPI[14] and also provides interactive API documentation based on Swagger UI[15], that can be seen in Figure 2.13 and Redoc[16]. Almost all of its features are developed following the RESTful API features [54].

For storing and preserving data, the emulator uses PostgreSQL which is a free and open-source relational database management system. The feasibility of administration and monitoring of the database is accomplished through pgAdmin, an open-source administration and development platform for PostgreSQL.

To store and preserve data, the emulator uses PostgreSQL[17], which is a free and open-source relational database management system. Administration and monitoring of the database are achieved through pgAdmin[18], an open-source administration and development platform for

[13]https://fastapi.tiangolo.com/
[14]https://www.openapis.org/
[15]https://swagger.io/tools/swagger-ui/
[16]https://redocly.com/redoc/
[17]https://www.postgresql.org/
[18]https://www.pgadmin.org/

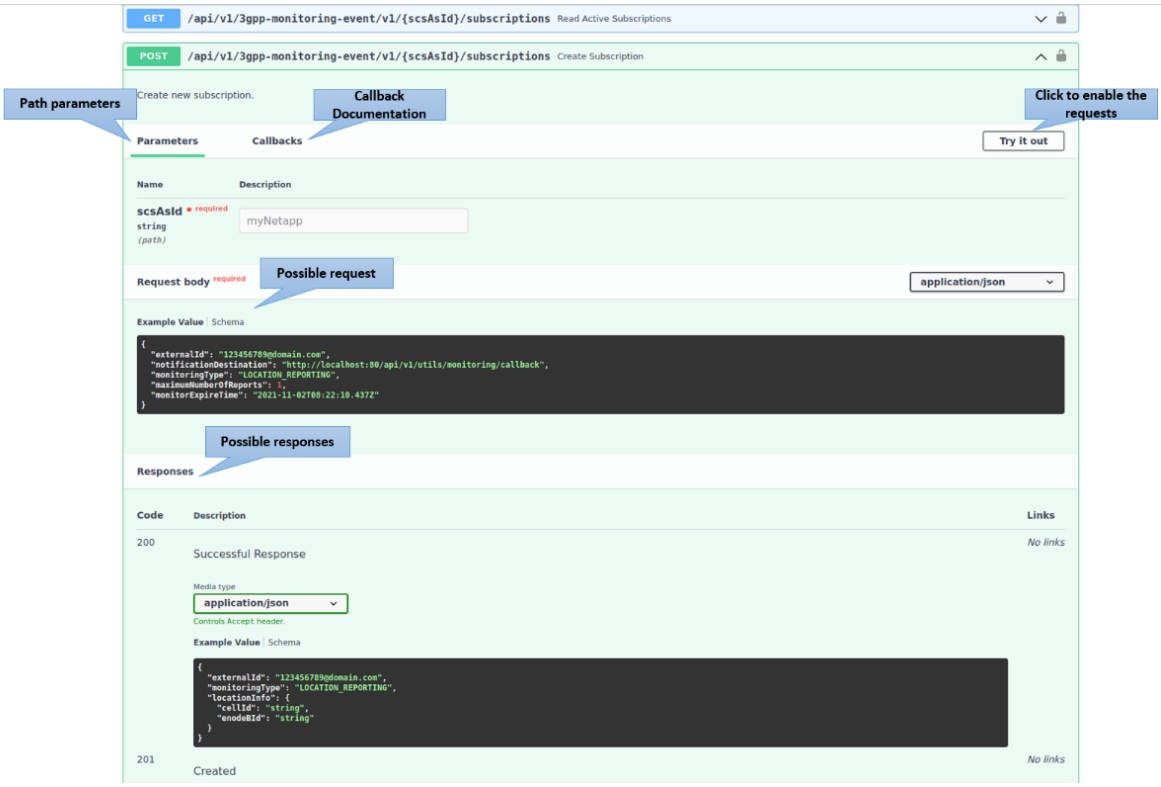

Figure 2.13: EVOLVED-5G NEF Emulator Swagger UI [54]

PostgreSQL.

The Dashboard displays details regarding the simulation's components, such as UEs, Cells, and predefined paths, which have been generated by a specific user. The Dashboard is presented in Figure 2.14.

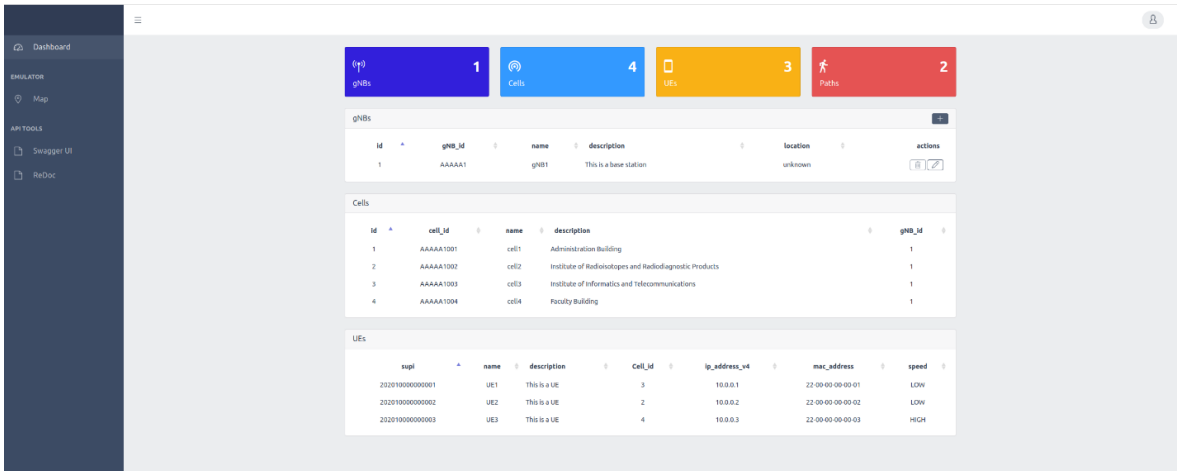

Figure 2.14: EVOLVED-5G NEF Emulator Dashboard [54]

For the simulation of the network a configurable set of 5G small cells are used. The details of the cell include cell's unique identifier, static geolocation and the coverage radius. Moreover, base stations are implemented to connect all the cells. It is important to note here that, the network does not reproduce a real operator deployment. To this end, the network scenario can

be dynamically configured by any user. Specifically, users can create their own base stations, cells, UEs and predefined paths.

To simulate the network, a customizable collection of 5G small cells are used. These cells are characterized by their unique identifiers, fixed geolocations, and coverage radius. Additionally, base stations are integrated to interconnect all the cells. It's crucial to emphasize that the network doesn't replicate a real-world operator's deployment. Consequently, users have the flexibility to dynamically configure the network scenario. More specifically, users can generate their own base stations, cells, UEs, and predefined paths.

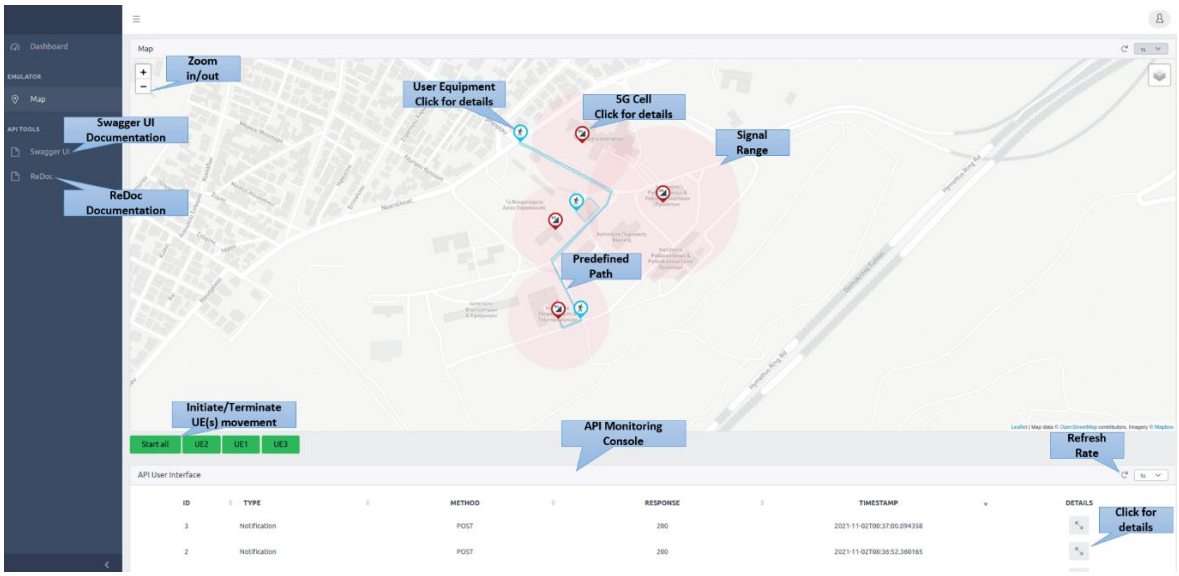

Figure 2.15: EVOLVED-5G NEF Emulator Map [54]

After the successful creation of the scenario for the emulation, the relevant information details are depicted in the Map page using OpenStreetMap[19]. At the bottom of Map page, there is an API User Interface that shows the monitoring of real time requests/notifications that the NEF APIs receive and generate. Users can experiment with NEF services either from the Swagger UI or directly through their application.

As previously mentioned, the primary objective of the verification process is to confirm the seamless communication between the Network Application and the 5G network. Consequently, the Network Application engages with the NEF Emulator to mimic this behaviour.

Three sets of tests are currently being defined, and can be seen in Figure 2.16. T1 and T2 refer to any communication initiated by the Network Application with synchronous responses from the 5G NPN infrastructure[57]. Here, services are exposed through the NEF emulator and CAPIF tool, utilizing NEF and CAPIF APIs respectively. T3 focuses on verifying asynchronous communication between the Network Application and 5G services, initiated by the 5G network and implemented through appropriate callback functions within the Network Application. For T3, the relevant Network Application REST API is put into test[57]. In the case of T1 and T2, verification tests are defined by exposing part of the NEF/CAPIF API

[19]https://www.openstreetmap.org/

invocation calls as REST APIs, enabling integration tests between the Network Application and 5G components. Alternatively, when this approach is not desirable, the relevant tests for the corresponding API calls are defined within the Network Application. In practice, the verification tests for a Network Application are established as unit tests between the Network Application and the exposed NEF and CAPIF APIs [57].

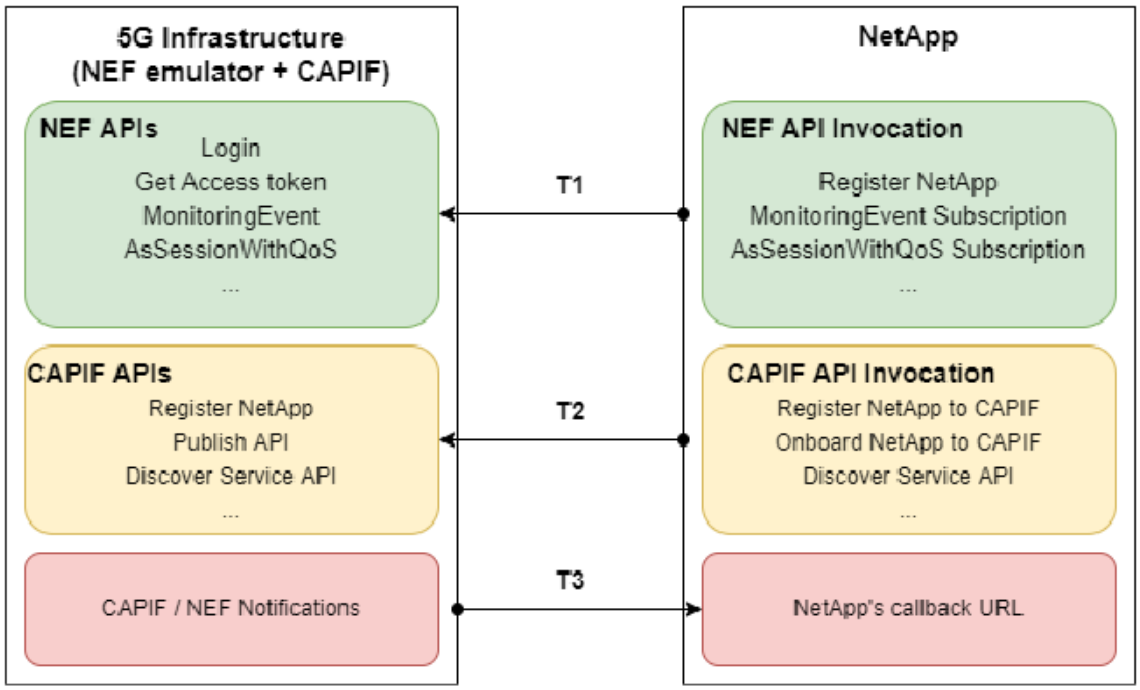

Figure 2.16: Network Application tests [57]

2.8.3 5GMediaHUB

The 5GMediaHUB project, funded by H2020 (ICT-41-2020), aims to speed up the testing and validation process of 5G-enabled media applications and Network Applications developed by third-party experimenters. This will be achieved through a facility that offers an open, integrated and well-equipped experimentation environment [58].

To accomplish this, 5GMediaHUB presents an architecture where its main components are (i) the Experimentation Tools layer, (ii) the NetApps & Slice layer, (iii) the 5G infrastructure layer (iv) and the UE layer. Fig. 2.17 depicts 5GMediaHUB's architecture [59].

Regarding the validation of 5G services, 5GMediaHUB has defined KPIs divided in three groups, (i) the Core 5G KPIs that are related to the 5G network performance, (ii) the Service KPIs that will be leveraged to quantify the performance of media applications when deployed at 5G infrastructures (iii) and the Platform specific KPIs, to quantify the effect of the experimentation facility in reducing the service creation and experiment setup times [60]. Each KPI is associated with a baseline value and a target value, which correspond to what was possible before and after leveraging 5G technologies and innovations employed within the project [61].

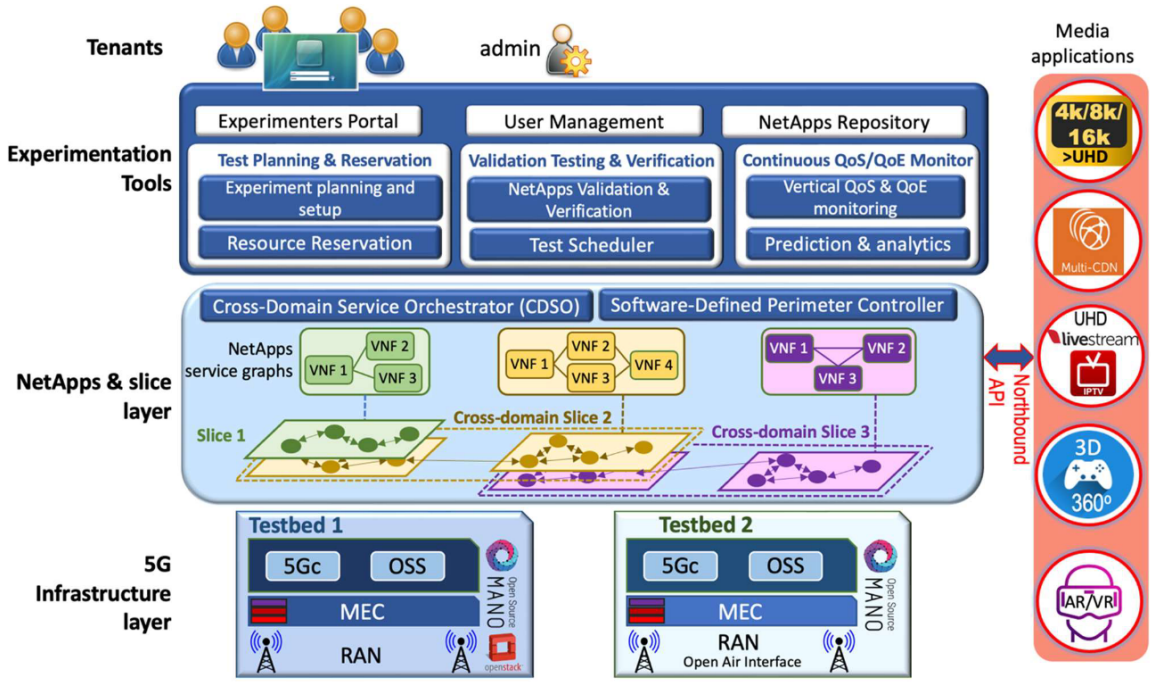

Figure 2.17: 5GMediaHUB Logical Architecture [59]

Experimentation Tools Layer

The Experimentation Tools Layer is composed by several components that are depicted in Fig. 2.18. These components are [59]:

- **User Management Module** - responsible for the authentication and authorization of all other modules in a single-sign on way;
- **Experimenters Portal** - user interface of the experimenter for setting up an experiment and visualizing the data KPIs;
- **Test, Planning & Reservation Engine** - engine responsible for reserving all the necessary resources for running an experiment;
- **Validation, Testing & Verification Engine** - in charge of validating if the Network Applications meet the required KPIs and Service Level Agreement (SLA) guarantees;
- **Continuous QoS/QoE Monitoring Engine** - engine for calculating the necessary KPIs based on the data collected by the experimenter's portal. KPIs are then exposed through REST APIs to 3rd party applications for use by the application developers;
- **NetApps Repository** - enables the developer to design and build Network Applications, reuse already existing ones for extending their use and finally onboard them on the Cross-Domain Service Orchestrator (CDSO);
- **Data Collector** - module responsible for collecting the data and KPIs from the respective data sources, making them uniform, calculating the necessary KPIs and storing the data in the database.

The Network Applications need to include (i) the advertised KPIs, allowing the workflow to check if they've been met, (ii) the Jenkinsfile that will describe the steps of the automated

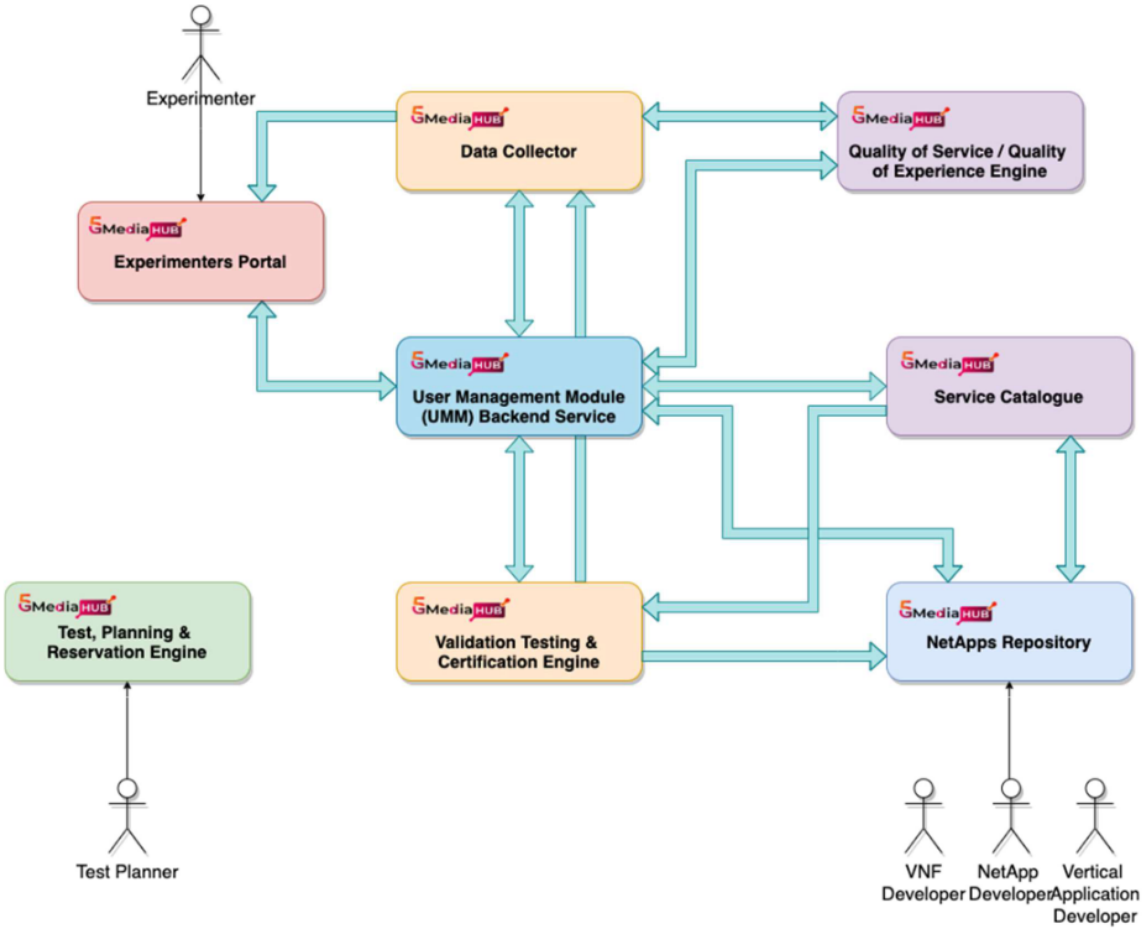

Figure 2.18: Overall Experimentation Tools Layer Architecture [59]

pipeline (iii) and one or more test applications that will utilise the Network Application and will run during the workflow [61]. The Validation and Verification Engine will evaluate on the test results by comparing the KPIs measured during the tests with the advertised KPIs from the Network Applications and the results will be either Pass or Fail, then they will be propagated to the data collector and the Netapps Repository[61].

Network Applications

The project strongly focuses on the media vertical sector, catering to applications that involve extremely high-resolution video streaming (4K/8K) with elevated frame rates. It also includes Augmented Reality (AR), Virtual Reality (VR), and Extended Reality (XR), along with 360-degree immersive experiences, and more [62].

Having mentioned that, 5GMediaHUB has outlined three distinct usage cases, each with multiple scenarios. The first use case relates to AR, VR, and XR applications, aiming to provide a foundation for immersive technologies. The second case involves Smart Media Production, designed to guarantee and assess the SLA compliance of these applications. Lastly, the third case is centered around Smart Media Content Distribution, intended to experiment with cross-domain media distribution through the Experimentation Facility, while

also evaluating its impact on QoS and Quality of Experience (QoE) [60], [63].

Immersive 360º VR Media Experiences

Taking into account the extensive range of KPIs established by 5GMediaHUB, the Immersive 360º VR Media Experiences scenario, which falls within the ambit of the first use case, will serve as example.

In this setting, individuals using VR headsets have the opportunity to fully engage with a virtual environment, enabling them to explore their surroundings in a 360º manner. The system configuration for this particular scenario facilitates the presentation of remote speakers and their audience within a virtual conference setting [60].

Given the focus on video quality and transmission in this scenario, the upcoming KPIs mostly concern video quality and the transmission of videos across 5G networks. So for this case, 5GMediaHUB has defined to measure the following set of KPIs:

- **Motion-to-Photon (MTP) Latency** - measures the time between the movement of the user, since they are using VR helmets, and the video content that is displayed.
- **Rendered Video Quality** - measures the video quality for the user.
- **Missing display area** - measures the maximum percentage of pixels without contents under normal user behaviors.
- **Application Latency** - measures the full time where a position request from the headset is answered with a new image buffer.
- **Frame Error Rate** - measures the percentage of video frames with errors.
- **Video jitter** - measures the maximum variation on the video frame rate measured over time.
- **Video Resolution** - measures the total amount of pixels transmitted per frame in adaptive streaming scenarios.
- **Peak throughput Downlink and Network Bandwidth Downlink** - measures the peak throughput and network bandwidth downlinks.
- **Reliability and Availability** - measures if the application works as expected and its uptime and accessibility.

2.8.4 5GASP

The 5GASP project aims at shortening the idea-to-market process through the creation of a European testbed, fully automated and self-service. Enabling SMEs to develop Network Applications, test them under close-to-real conditions and certify them before they finally reach their targeted Network Operators and/or Vertical industry customers [4], [64].

This dissertation surges in the context of 5GASP and the more important outline in it, taking into the account the work that will be implemented, is the CI/CD pipeline which will be described in the following paragraphs.

5GASP has a CI/CD service where when a Network Application is onboarded on the Portal, the CI/CD pipeline is triggered initializing the tests' execution. These tests are being

monitored and when finished the testcases results are sent back, validating the Network Application.

In detail, the onboarding and deployment process of a Network Application is triggered by the interaction of the developer with the 5GASP Portal. 5GASP enables the onboarding of a bundle containing the Network Application and a Testing Descriptor, which is a YAML Ain't Markup Language (YAML) file that allows developers to provide specifications of the tests they wish to perform. Once the information is onboarded to the Network Application Onboarding and Deployment Service (NODS), the CI/CD Service is triggered. This service, which is illustrated in Fig. 2.19, is composed by some core elements [64]–[66]:

- **CI/CD Manager** - Is the entry-point for the CI/CD Service. It is available via a REST API, implemented using FastAPI, and is triggered by the NODS. After the Network Application's deployment, the NODS submits a new request to the CI/CD Manager, which will create a configuration file to be submitted to the CI/CD Agents located on the testbeds where the Network Application was deployed. The CI/CD manager also stores the results on the Results Repository. When the validation process ends, the CI/CD Manager is responsible for informing the NODS so the developers can get the results.
- **CI/CD Agents** - Are responsible for performing the tests on the Network Applications. Each testbed has a CI/CD Agent deployed on the Network Slice where the Network Applications will be deployed, and it's registered in the CI/CD Manager, so it knows how to communicate with it. The Agents receive the configuration file from the CI/CD Manager, detailing the testing process. They are responsible for triggering the metrics collection mechanisms in the Network Applications, gathering the tests from the LTR, performing them on the Network Applications, and informing the CI/CD Manager of the results of the tests which will be stored in the Results Repository.
- **Local Test Repository** - The LTRs store all the tests that can be performed in a facility and make them available for the CI/CD Agents to gather and execute them on the Network Applications.
- **Test Results Visualization Dashboard** - The TRVD enables the Network Application developers to get the outcomes and outputs of the testing and validation phase. It will present the outputs and results stored in the CI/CD Manager's Results Repository.

Regarding the tests performed on the Network Applications, two types of tests are defined: predefined-tests and developer-defined tests. Predefined-tests are already onboarded in the 5GASP ecosystem so that all the developers can use them to validate global aspects, such as communication latencies or the bandwidth of the testbed. Developer-defined tests are onboarded by the Network Application developers and address the behavior and singular aspects of the Network Application they developed [67].

For a more comprehensive understanding of the 5GASP architecture, refer to Fig. 2.20 where it is illustrated.

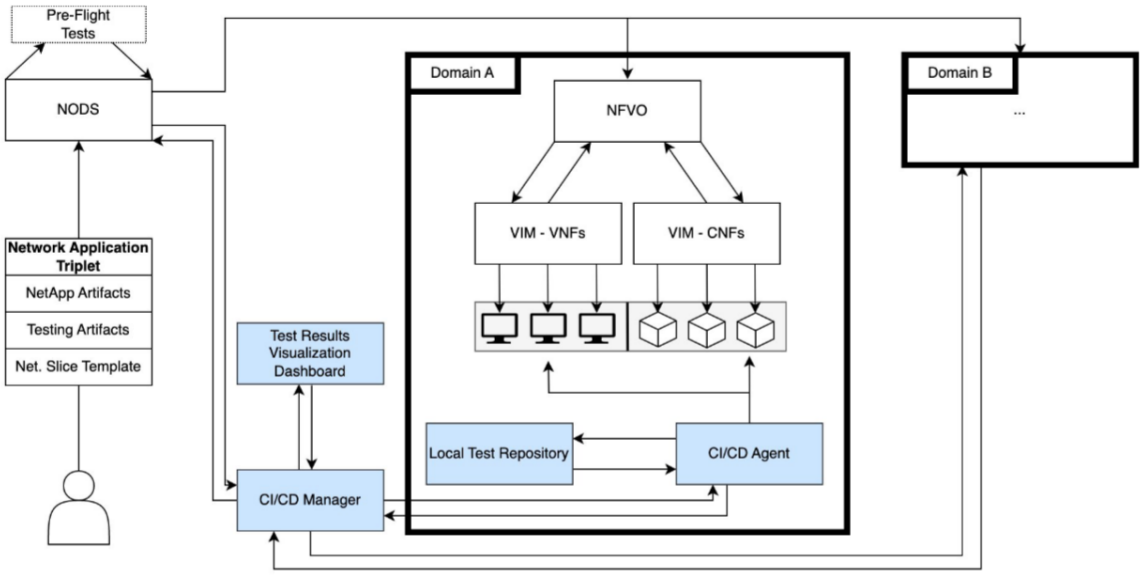

Figure 2.19: 5GASP's CI/CD Service High Level Architecture [67]

5GASP has several facilities, which are the Aveiro, Bucharest, Patras, Bristol, Ljubljana and Murcia facilities. All these facilities are connected to the 5GASP Portal and to the NODS. In the Aveiro facility is where the OSM release 12 is deployed.

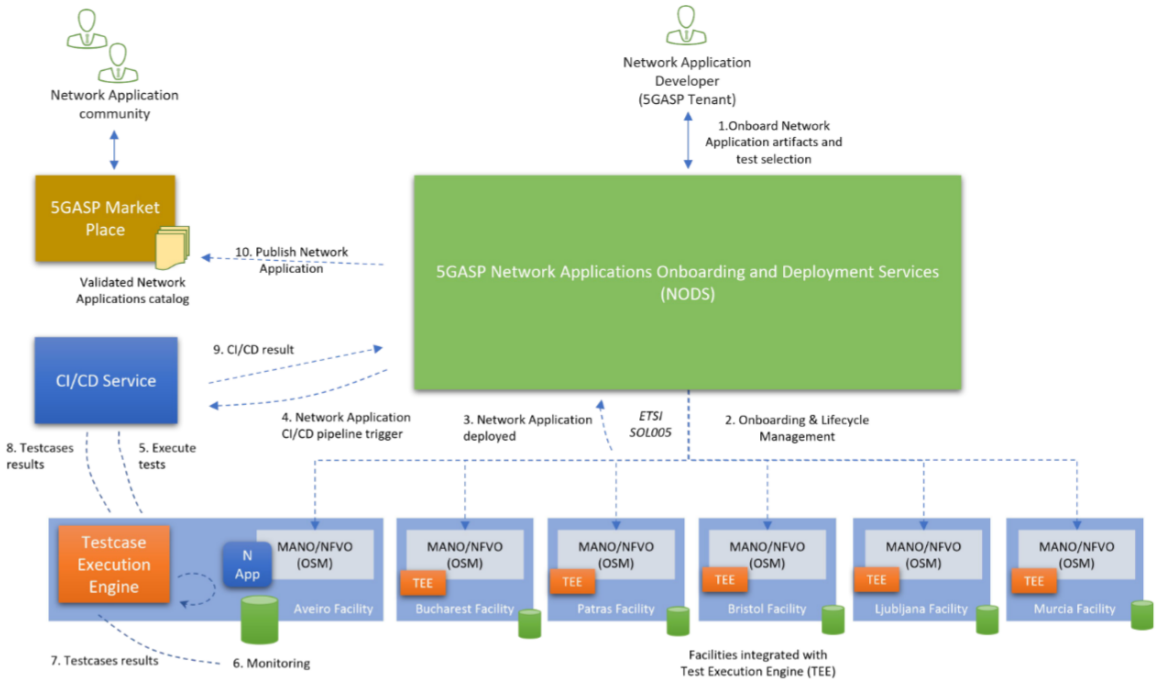

Figure 2.20: 5GASP Architectural Workflow [67]

2.9 Work Proposal

Now that the background concepts were introduced, a work proposal must be conceived. Thus, this section presents the problems addressed by this dissertation and presents a work proposal to solve them. As mentioned before, this dissertation's goals fall within the scope of the 5GASP project.

As previously stated Section 2.8.4, 5GASP aims to provide a fully automated and self-service testbed, which shortens the idea-to-market time. To do so, 5GASP has already implemented a testbed featuring a CI/CD pipeline for conducting tests to validate Network Applications. A set of pre-defined tests has also been developed to test mainly the VNF and NS packages, covering elements such as the VNF and NS descriptors, the VNF Juju Charms, among others.

Nevertheless, Network Applications need further validation in order to become certified and to be ready for deployment in a real 5G infrastructure. This falls into the two objectives of this dissertation: (i) the creation of additional tests designed to validate Network Applications in accordance with the 5GASP testing scopes and consequently (ii) the expansion of the NEF emulator to enable the creation of tests that take advantage of it.

Furthermore, the additional tests defined in this dissertation will mainly fall into the 5G Readiness Testing Scope defined by 5GASP. This scope purpose is to validate the communication between the Network Applications and the 5G Network, thus the need for the NEF, which is one of the components of the 5G architecture and not only provides extensive network information but also enables network changes to be requested. Hence, it makes it highly attractive to Network Applications in terms of interacting with the 5GC, as the interaction process is very straightforward. This functionality will be provided through the NEF Emulator.

Regarding the emulator, Northbound APIs will be added to expose a wider range of network capabilities to the Network Applications for them to consume. The NEF emulator will serve as the backbone for the tests developed in this dissertation. However, part of the validation phase will require the logging of the used endpoints, to validate the interaction with the Network Applications. To achieve this, an API will be developed along side the NEF emulator. Additionally, it is essential to integrate the NEF emulator into the existing CI/CD pipeline.

In addition, from the tester's point of view, a Network Application is a black box. In other words, there is no visibility or control over the internal operations of a Network Application. However, the tester needs to control the flow of actions to avoid running tests immediately upon startup, since essential components required for testing may not be available. To solve this problem, there is a need for an additional API that will be provided as a standard to all Network Applications, enabling them to define the logic for initiating interactions between Network Applications and the NEF emulator.

Chapter 3 addresses the general architecture, while Chapter 4 addresses the implementation of all the elements and tests developed.

2.10 Chapter Summary

This chapter begins by highlighting the challenges encountered in traditional networks, including rising costs attributed to their lack of adaptability and flexibility in meeting evolving network demands.

Section 2.2 introduces the NFV paradigm, proposed by ETSI. NFV enables the separation of NFs from their physical infrastructure, providing greater flexibility, scalability, and faster deployment capabilities. These advantages lead to a reduction in time-to-market.

In order to facilitate the adoption of the NFV paradigm, ETSI introduced the MANO framework, which is detailed in Section 2.2.1. This framework plays a crucial role in managing and orchestrating NFs, with its primary components being the NFVO, the VNF Manager and the VIM. The MANO framework got widely adopted, leading to the appearance of numerous implementations. Sections 2.3.1 and 2.3.2 provide insights into some of the most successful implementations, with OSM standing out.

Section 2.4 introduces the concept of a Network Application, which has gained acceptance within the 5G community. It refers to a collection of services deployed over 5G infrastructures that simplify the implementation and deployment of vertical systems on large scale.

The 5GC System Architecture, specified by 3GPP is introduced in Section 2.5 where the NFs that compose it are described. Furthermore, the NEF is referenced in more detail in Section 2.5.1. NEF provides secure means for exposing capabilities provided by other 5GC NFs to external systems.

Sections 2.6.1 and 2.6.2 present the concepts of DevOps and CI/CD, respectively. In Section 2.6.3, these two concepts are integrated within an NFV environment, offering a more smooth transition to NFV environments.

Section 2.7 presents several Testing Automation Frameworks and dives deep in the Robot Framework, which was used in this work.

Section 2.8 presents other works developed in the context of this dissertation. Section 2.8.1 presents the VITAL-5G project, which offers a Testing & Validation Framework, and introduces the Testcase template that serves to specify the test details, configure network elements, and define metrics and KPIs. This project also introduces a way for defining metrics to validate Network Applications. Section 2.8.2 presents EVOLVED-5G, which introduces the CAPIF Tool, the NEF Emulator and an architecture to validate and certificate Network Applications that use the emulator to validate some aspects of the interaction with the 5G infrastructure. 5GMediaHUB is presented in Section 2.8.3 and focuses on Media Applications. The architecture for the validation is presented and an example of one Network Application and its KPIs are shown. The last project referenced is 5GASP, in Section 2.8.4, which is the project where this dissertation is inserted. The overall architecture of the project is presented alongside a detailed explanation of the CI/CD Pipeline.

Lastly, Section 2.9 presents the work proposal of the dissertation, where the two main objectives are discussed: (i) the creation of additional tests designed to validate Network Applications and consequently (ii) the expansion and integration of the NEF emulator into

the 5GASP CI/CD pipeline to enable the creation of tests centered on it.

CHAPTER 3

Architecture

Section 2.9 presented the work proposal for this dissertation. Within that chapter, two distinct objectives were outlined: (i) the development of tests designed to validate Network Applications and (ii) the extension of the NEF emulator to enable the creation of tests that utilize it.

The purpose of this chapter is to introduce the architecture that has been devised to accomplish the two aforementioned objectives.

First it is presented how the proposed components, namely the NEF emulator, the Report API and the Mini API interact with each other. Afterwards, an architecture is proposed integrating them on the 5GASP CI/CD pipeline.

3.1 Problem Statement

Network Applications must undergo validation to guarantee they meet the anticipated quality and performance criteria, which are determined by both the application's intended purpose and the environment in which it will be used.

Throughout the course of this dissertation, the 5GASP project established various certification criteria with varying levels of priority. Among these, the criteria identified as mandatory serve as the fundamental requirements for certifying a Network Application. In the context of this dissertation, the following mandatory criteria hold the utmost significance [68]:

- Should follow the NFV model [7].
- Should follow relevant 3GPP security definitions and recommendations.

Taking into account all the project's definitions, five testing scopes were defined: (i) 5G Readiness Testing, (ii) Security & Privacy Testing, (iii) Performance & Scalability Testing and (iv) Availability & Continuity Testing [68].

Considering the previously mentioned definitions, the 5G Readiness and Security & Privacy testing scopes carry greater significance and must be fulfilled by the Network Applications

in order for them to be certified by the project. Thus, this dissertation focuses on these two scopes.

Another aspect to consider is that, as previously discussed in Chapter 1, there is currently no actual implementation of NEF. The existing components consist solely of the standards defining Northbound APIs. Additionally, the standards for Southbound APIs, which facilitate communication between the NEF and the NFs within the 5GC, are in their early developmental stages. Nevertheless, the EVOLVED-5G project has developed an emulator for the NEF, which includes the implementation of two APIs, the MonitoringEvent API and the AsSessionWithQoS API based on the TS29.122 [55] standard. This emulator enables the emulation of Southbound behavior and offers the opportunity to incorporate additional Northbound APIs to expose further capabilities.

Regarding the testing scopes, 5G Readiness entails certifying the Network Application against the 5G Network, involving rigorous testing of its communication and interaction capabilities. This is where the NEF emulator plays a crucial role, as NEF exposes the 5G Network's capabilities through Northbound APIs, allowing the simulation of communication between Network Applications and the 5G Network.

The first proposal of this dissertation, consists of extending the Northbound APIs of the NEF emulator, according to the standards, and to offer more network capabilities for external applications to consume. Additionally, in the context of this work, they will serve to validate if a Network Application follows the correct way of communicating with the 5G infrastructure. To achieve this, Network Applications must make requests to the Northbound APIs, and these interactions should be recorded for validation in the final stage.

The initial question that this raises is how to log the requests. The answer was to develop an API, the Report API, that offers the possibility to create, delete, and retrieve a shared file between the Report API and the emulator. Using this solution, the NEF emulator, through an implemented decorator, intercepts requests made to the Northbound APIs. It captures both the request and the corresponding response, and then records this information in the shared file previously mentioned. This record includes details such as the names of the endpoints invoked, the methods employed, the provided parameters, the transmitted payload, the response's status code, and its associated message, among other information. A high level representation of what was just explained is presented in Figure 3.1

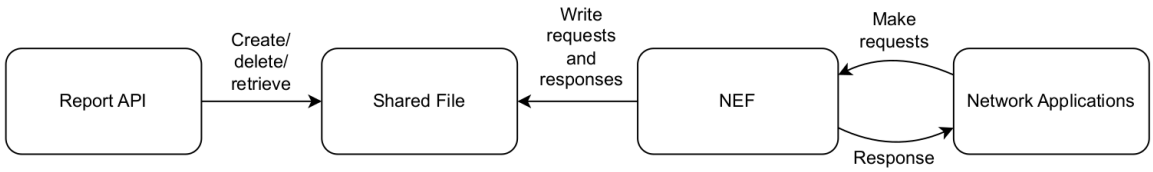

Figure 3.1: High Level Interactions with the Shared file

On the other hand, as stated in Section 2.9, Network Applications are viewed as black boxes by testers, meaning that their internal code is unknown. This lack of control over the course of Network Applications' actions can pose challenges in controlling the flow of the

testing pipeline, as Network Applications may require a component that is not yet available. The solution to avoid this issue, is to provide to the Network Application owners an API, the Mini API, that will be used as a standardized interface for all Network Applications, offering endpoints for them to define the logic to trigger interactions between Network Applications and the NEF emulator. By adopting this approach, testers can exercise control over the timing of communications to prevent conflicts with other pipeline components.

Once these components have been successfully integrated into the 5GASP CI/CD pipeline, it becomes possible to create tests aimed at assessing the 5G Readiness of Network Applications.

3.2 Architecture and Specifications

The previous section listed the core components that will be used for the testing pipeline. To summarize, these components are the following: (i) the Mini API, (ii) the NEF emulator, (iii) the Report API, and (iv) the CI/CD Agent. A description of each component is presented in the following subsection.

3.2.1 Components

Mini API

The Mini API is a REST API that has been developed using FastAPI. It aims to control the flow of operations between the Network Applications and the 5G infrastructure. This API is offered as a standard to all the Network Application owners, which are responsible to develop it alongside their Network Applications in order for them to be validated within the 5G readiness scope.

NEF Emulator

The NEF emulator is the most important component of this proposal. Its primary aim is to make 5G capabilities accessible to Network Applications by emulating a real-world 5G NEF. These capabilities are made accessible through standardized Northbound APIs, which are accessible via a REST API. Within the validation method proposed, the NEF emulator also has the function to log the endpoints and associated information that are invoked by Network Applications communicating with it. Additionally, the NEF emulator has the capability to produce values for a particular API and its corresponding parameter. This functionality is intended to reduce the reliance on placeholder values for the Network Application to utilize.

Report API

The Report API, also developed using FastAPI, serves to create, delete and retrieve the shared file that exists between this API and the NEF emulator. The shared file is a JavaScript Object Notation (JSON) file created to store the information about the communication that takes place between the Network Application and the NEF Emulator, concerning the requests made to Northbound APIs. Lastly, this file is then gathered and used on the validation process,

leveraging the logged data within to evaluate whether the Network Application follows correct communication practices.

CI/CD Agent

The CI/CD Agent was not a component developed in this dissertation, it already existed within the 5GASP project, as described in Section 2.8.4. The CI/CD Agent is responsible for collecting tests from the LTR, running them, and transmitting their outputs and results to the CI/CD Manager. Through the tests that are executed by the CI/CD Agent, the actions to trigger the Report API and the Mini API are done.

3.2.2 Modules' Interaction

The initial and crucial stage for this proposal involved defining how would each component interact with one another. This is depicted in Fig. 3.2.

Throughout the Network Application development process, the Network Application owners are responsible for implementing the logic of their Mini API endpoints to facilitate testing, also they need to provide a Testing Descriptor that provides the testcases for the Network Application to be tested.

Next, when the validation phase is reached, the Report API is activated to generate a report file, which is a shared file with the NEF emulator to record the interactions made with the emulator. Afterwards, the configuration of the Mini API is initiated, involving the setup of parameters such as the emulator's IP address, ports, credentials, and more. Following the

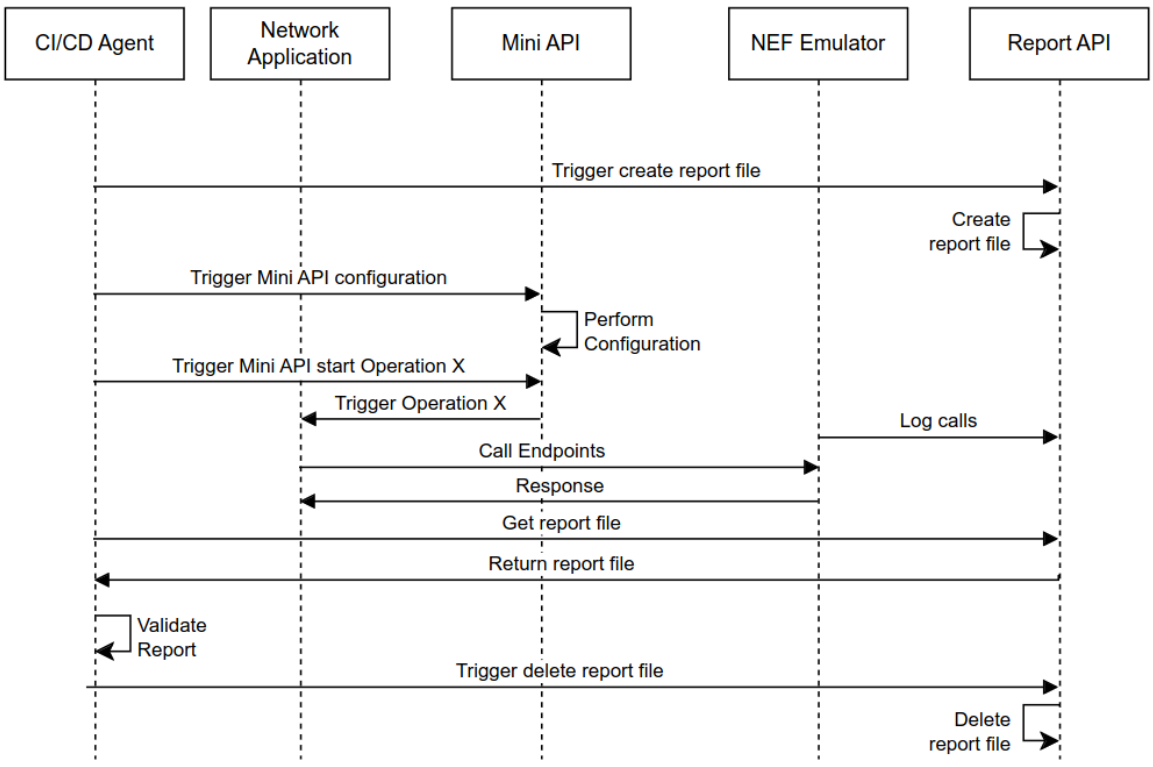

Figure 3.2: Components' Interaction

configuration, the initiation of an operation becomes possible. This initiation is signaled, through the tests, to the Mini API, which then in turn signals the Network Application to trigger the desired operation. The Network Application subsequently calls the NEF emulator endpoints associated with that operation. During this phase, the emulator logs the call and the response that is sent back to the Network Application. Finally, to complete this cycle, it is necessary to retrieve the report through the Report API. In the final stage of the tests the Network Application is validated with the retrieved report, and once this is done, the report is deleted.

3.3 Testing Pipeline

After introducing the main components proposed in this dissertation, the next step is to integrate them into the 5GASP CI/CD pipeline, which was already described in Section 2.8.4.

Fig. 3.3 illustrates the proposed architecture.

The proposed architecture incorporates the NEF emulator and the Report API into the Aveiro Facility. The NEF emulator is responsible for making standardized APIs available to Network Applications, simulating events, and logging the interactions with it. The Report API manages a shared file for logging purposes.

The Mini API is placed alongside the Network Application to trigger the interactions made to the NEF emulator by the Network Application.

The proposed flow of the testing pipeline is as follows:

1. Onboarding of the Network Application with the Testing Descriptor into the 5GASP Experimentation Service.
2. The 5GASP Experimentation Service triggers the CI/CD pipeline.
3. The CI/CD Manager triggers the CI/CD Agent to execute the tests.
4. Execution of the tests by the CI/CD Agent, which include the creation of the report file, the initiation of the Mini API, the retrieval of the report, and its subsequent validation.
5. The validation is done and the results are propagated back to the 5GASP Experimentation Service.

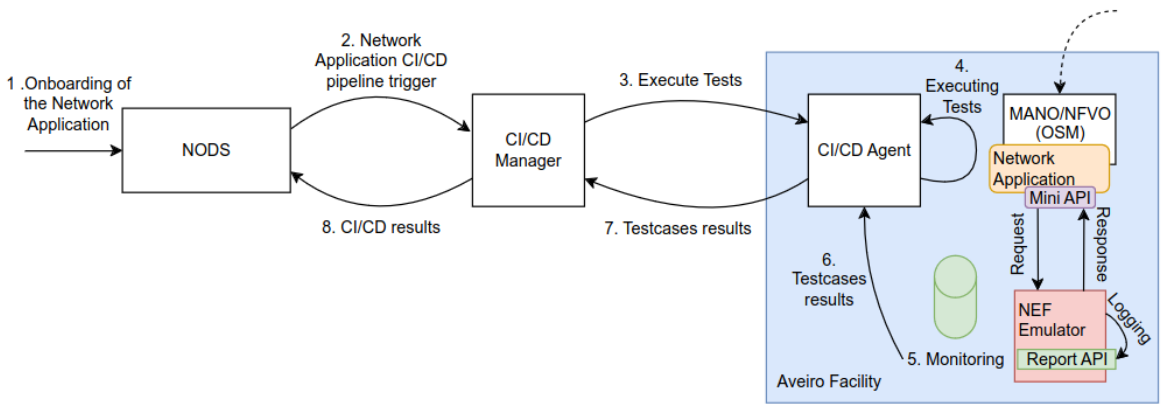

Figure 3.3: Proposed Architecture

3.3.1 Report File

To have a better view of the validation process, it is essential to understand the structure of the report file. In summary, the report file is a shared file controlled by the Report API. Within this file, the NEF Emulator records the requests sent to it by the Network Applications and their subsequent responses.

For example, a Network Application that authenticates using the NEF Emulator and then subscribes using the MonitoringEvent API will have an associated report file with the expected contents as shown in Code Block 3.1.


```
[
    {
        "id": 1,
        "endpoint": "/api/v1/login/access-token",
        "method": "POST",
        "request_body": {},
        "scsAsId": "napp",
        "afId": null,
        "subscriptionId": null,
        "transactionId": null,
        "configurationId": null,
        "provisioningId": null,
        "setId": null,
        "nef_response_code": 200,
        "nef_response_message": "{'access_token':'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9',
        ↪  'token_type':'bearer'}"
    },
    {
        "id": 2,
        "endpoint": "/nef/api/v1/3gpp-monitoring-event/v1/napp/subscriptions",
        "method": "POST",
        "request_body": {
            "externalId": "10001@domain.com",
            "notificationDestination":
            ↪  "http://localhost:80/api/v1/utils/monitoring/callback",
            "monitoringType": "LOCATION_REPORTING",
            "maximumNumberOfReports": 1,
            "monitorExpireTime": "2023-03-09T13:18:19.495000+00:00",
            "maximumDetectionTime": 1,
            "reachabilityType": "DATA"
        },
        "scsAsId": "napp",
        "afId": null,
        "subscriptionId": null,
        "transactionId": null,
        "configurationId": null,
        "provisioningId": null,
        "setId": null,
        "nef_response_code": 200,
```

```
        "nef_response_message": "{'monitoringType':'LOCATION_REPORTING',
        ↪  'externalId':'10001@domain.com' ,'ipv4Addr':'10.0.0.1',
        ↪  'locationInfo':{'cellId':null, 'gNBId':null}}"
    }
]
```

Code Block 3.1: Report file example

The first dictionary in Code Block 3.1 corresponds to the information associated with the login request, while the second dictionary is related to the monitoring subscription request.

In general , each entry is expected to consist of several key elements. These elements include an identifier used for numbering entries, the requested endpoint, the method used, any applicable request body, and either the *scsAsId* or *afId* for identifying the Network Application responsible for the request. Just on of these identifiers have a value ,t he choice between them is determined by the specific Northbound API in accordance with the standards.

Furthermore, the entries contain the *subscriptionId*, *transactionId*, *configurationId*, *provisioningId*, and the *setId*. The values associated with these parameters vary depending on the Northbound API and method used. For instance, one of them will possess values when executing a *GET* operation for a specific subscription given that they are mutually exclusive.

Lastly, the entry includes the *nef_response_code* and the *nef_response_message*. The values of these two fields, once again, depend on the Northbound API utilized and the response provided by the NEF Emulator.

3.4 Chapter Summary

This Chapter starts by introducing the problem statement, in Section 3.1, which dives into the scopes of the tests that are present on the 5GASP project. It presents the two scopes that will have emphasis on this dissertation, the 5G Readiness and the Security scopes. Then the components that will be created, implemented and/or used to validate the Network Applications are introduced, which are the NEF Emulator, the Report API, the Mini API, and the CI/CD Agent and their description is made in Section 3.2.1. Section 3.2.2 follows with the proposed interactions between them to support the validation of Network Applications.

Lastly, Section 3.3 integrates the components into the 5GASP CI/CD pipeline, enabling automatic validation of Network Applications and Section 3.1 presents an example of what to expect from a report file.

CHAPTER 4

Implementation

Chapter 3 presented the architecture proposed for this thesis. This Chapter presents the implementation of what was proposed. Section 4.1 focuses on the NEF Emulator, starting with an overview of the Northbound APIs added to the emulator. It proceeds to detail the Report Handler, which is responsible for recording the interactions between the Network Applications and the NEF emulator. Additionally, it discusses the endpoints introduced to support the information retrieved from the emulation and mentions the Generation of Values through the NEF Emulator, which enables the provision of more realistic values for consumption by Network Applications. Sections 4.2, 4.3, and 4.4 provide insights into the implementation of the Report API, the Mini API, and the Validation Tests, respectively.

4.1 NEF Emulator

4.1.1 Northbound APIs

As previous mentioned, in Chapter 1, at the time of this dissertation the only implementation of NEF is the emulator from the EVOLVED-5G project. The work developed was forked from this project. As stated in Section 2.8.2, the NEF-Emulator has already two APIs implemented, the MonitoringEvent API and the AsSessionWithQoS API from the TS29.122 [55] standard.

In order to assess the readiness for 5G compatibility, as previously mentioned, Network Applications must establish communication with the 5G Network. Subsequently, the next phase of this project involved determining the relevant APIs that needed implementation or addition. This decision was guided by the consideration of which parameters were pertinent to the Network Applications and could be supplied via their endpoints. This process relied on references such as TS29.122 [55] and TS29.522 [56], which encompass the standards for the Northbound APIs.

Table 4.1 presents an overview of the standardized Northbound APIs that were considered for implementation. It indicates whether they were added, not added, or partially added i.e. started but not concluded, in the NEF Emulator.

Northbound API	Standard - Version	Added
MonitoringEvent	29.122 - 17.8.0	Yes
AsSessionWithQoS	29.122 - 17.8.0	Yes
Resource Management of Background Data Transfer (BDT)	29.122 - 17.8.0	Yes
Chargeable Party	29.122 - 17.8.0	Yes
Non-IP Data Delivery (NIDD)	29.122 - 17.8.0	No
Device Triggering	29.122 - 17.8.0	No
Group Message Delivery via MBMS by MB2	29.122 - 17.8.0	No
Group Message Delivery via MBMS by xMB	29.122 - 17.8.0	No
Network Status Reporting	29.122 - 17.8.0	Yes
Communication Patterns (CP) Parameters Provisioning	29.122 - 17.8.0	Yes
Packet Flow Description (PFD) Management	29.122 - 17.8.0	Partially
Enhanced Coverage Restriction Control	29.122 - 17.8.0	No
Network Parameter Configuration	29.122 - 17.8.0	Yes
MSISDN-less Mobile-Originated SMS	29.122 - 17.8.0	No
RACS (Radio Capability Signaling) Parameter Provisioning	29.122 - 17.8.0	Yes
Traffic Influence	29.522 - 17.8.0	Yes
NIDD (Non-IP Data Delivery) Configuration Trigger	29.522 - 17.8.0	Partially
Analytics Exposure	29.522 - 17.8.0	Yes
5G LAN Parameter Provision	29.522 - 17.8.0	No
Applying BDT Policy	29.522 - 17.8.0	No
IPTV Configuration	29.522 - 17.8.0	No
LPI (Location Privacy Indicator) Parameter Provision	29.522 - 17.8.0	No
Service Parameter	29.522 - 17.8.0	No
ACS Parameter Provision	29.522 - 17.8.0	No
MO LCS Notify	29.522 - 17.8.0	No
AKMA	29.522 - 17.8.0	No

Table 4.1: Northbound APIs

The implementation of these APIs exclusively consists of the endpoints outlined in the standards, without integrating any interaction with the scenarios simulated by the NEF emulator. From the perspective of a tester, the emulator's values are irrelevant, what truly matters is the application's behavior and communication with the emulator. Consequently, only the endpoints are necessary to achieve the validation of the 5G Readiness on Network Applications.

4.1.2 Report Handler

To achieve the aim of this dissertation, which focuses on validating Network Applications, it was required to log the calls made from the Network Application to the NEF emulator. To accomplish this, a function was incorporated into the emulator, which activates upon each call made to it. The objective of this function is to document the invoked method and endpoint, capture the body and/or parameters provided by the Network Application, and record the response code and message generated by the NEF, among others. The Code Block 6.1, presented in the Appendix of this document, displays this function.

This is then incorporated into all the Northbound APIs developed to allow for the logging of their respective endpoints.

The log file is stored within a shared volume, as both containers running the NEF Emulator and the Report API operate on the same machine.

4.1.3 Emulation Endpoints

Throughout the course of this dissertation, 5GASP introduced several endpoints into the emulation segment, which were intended for use in the testing phase. Figure 4.1 displays these endpoints.

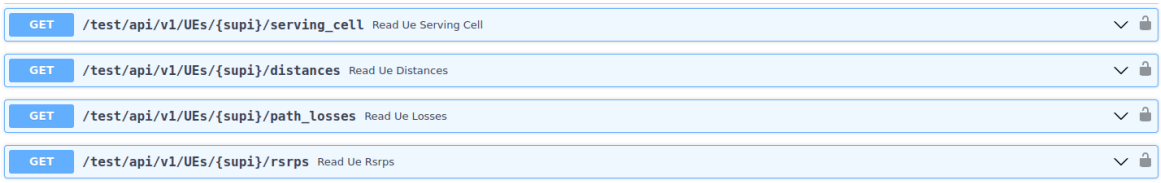

Figure 4.1: Endpoints added by 5GASP

Nonetheless, there was a need for an extra endpoint dedicated to identifying the occurrence of a handover. To fulfill this requirement, an endpoint was introduced to provide information regarding the cells involved in a handover event. This endpoint is presented in Figure 4.2.

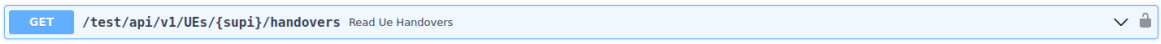

Figure 4.2: Developed Handover endpoint

4.1.4 Generating Values

While the values supplied by the NEF emulator are essentially placeholders, it became intriguing in the later stages of this dissertation to enhance the emulation aspect by introducing a mechanism for generating and utilizing values for specific API parameters through a broker. This effort aimed to move beyond mere dummy values and provide more realistic inputs to the Network Application.

As this decision was made relatively late in the process, it was only implemented for the AsSessionWithQoS API. This choice was based on the feasibility of obtaining values for specific parameters and their relevance to Network Applications.

This feature provides a default sinusoidal function capable of generating values for a designated parameter within the aforementioned API when requested. Users not only have the flexibility to customize this function, but also have the option to create and switch to their own custom functions for producing parameter values.

The default sinusoidal function implemented is described by the following formula:

$$f(x) = A\sin(Bx + C) + D$$

where A represents the amplitude, which is the maximum distance from the centerline to the peal of the wave, B is the angular frequency, which determines how quickly the function oscillates, C is the phase shift, which shifts the function horizontally along the x-axis and D is the vertical shift or offset, which shifts the function vertically along the y-axis.

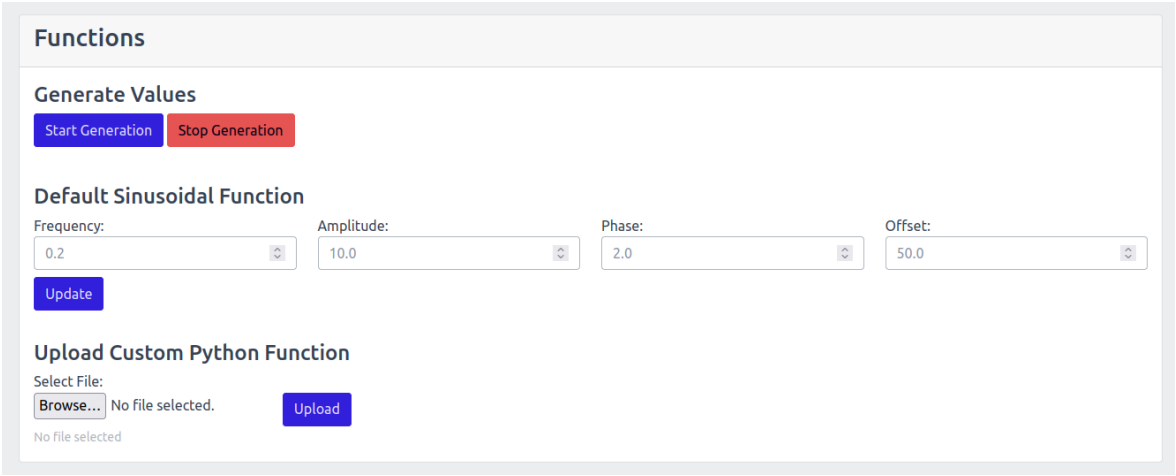

Figure 4.3: UI for the Generation of Values

AsSessionWithQoS API the tests are automated, they cannot be initiated through the UI. Therefore, it was necessary to create additional endpoints to facilitate this capability, as illustrated in Figure 4.4.

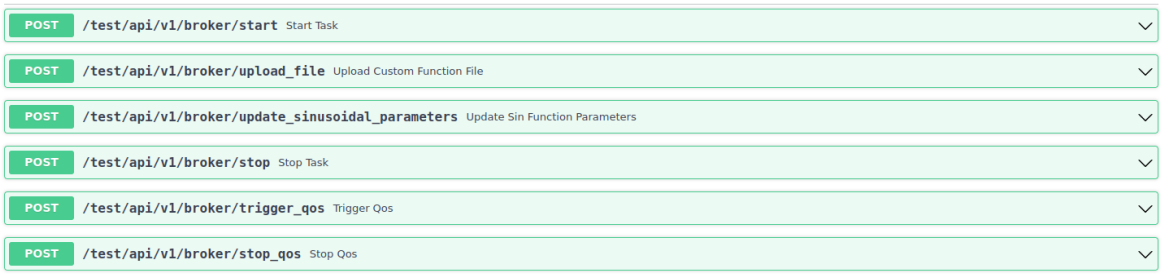

Figure 4.4: Endpoints that enable the Generation of Values

It is crucial to clarify how a Testing Agent can make use of this function. First, it it required to initiate the default production of values to the broker through the */test/api/v1/broker/start* endpoint. Once this is done, if necessary, adjustments to the sinusoidal coefficients can be made using the */test/api/v1/broker/update_sinusoidal_parameters* endpoint, or the entire

function responsible for generating values can be replaced by providing a Python file through the */test/api/v1/broker/upload_file* endpoint. As mentioned before, this method only works for the AsSessionWithQoS API, so the next endpoints used are dedicated specially to this API. The */test/api/v1/broker/trigger_qos* and the */test/api/v1/broker/stop_qos*, respectively, signal the the AsSessionWithQoS API to start and stop consuming the values from the broker. When this behaviour is triggered, it also indicates which parameter of the API is being updated. If dealing with a nested parameter, it should be separated with a hyphen, as in the case of *usageThreshold-uplinkVolume*. To stop the broker from generating values exists the */test/api/v1/broker/stop* endpoint.

To use the values generated by this functionality, an AsSessionWithQoS subscription needs to be done, using the */api/v1/3gpp-monitoring-event/v1/scsAsId/subscriptions POST* method. From this, it is generated a unique identifier known as *subscriptionId* and with it, the *GET* method from the AsSessionWithQoS API to retrieve specific information about this *subscriptionId* can be called using the */nef/api/v1/3gpp-monitoring-event/v1/scsAsId/subscriptions/subscriptionId* method. This method will provide details about the requested *subscriptionId*, which will be updated with the values obtained from the broker if its usage was previously signaled as described before.

It's important to mention that this method has two limitations: (i) it only works for the AsSessionWithQoS API, as previously stated, (ii) it can only be used on one parameter at a time due to its implementation, and (iii) it only works on numeric parameters.

4.2 Report API

To tackle the problem discussed in the Section 2.9, an API, named Report API, was developed. It was implemented using the FastAPI [1] framework. This REST API is straightforward, comprising only three endpoints, as depicted in Figure 4.5.

The API is encapsulated within a Docker Image and deployed alongside the NEF emulator through Docker Compose. Its sole purpose is to manage the creation, deletion, and retrieval of the report file, which serves as a shared file between the NEF Emulator and this API.

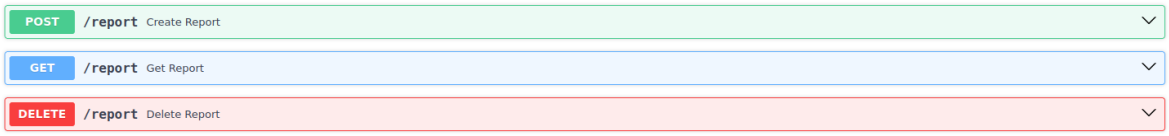

Figure 4.5: Report API endpoints

4.3 Mini API

The Mini API was developed using FastAPI. The aim of the API is to serve as a wrapper to trigger the interactions between the Network Application and the NEF emulator. However, it's important to note that as of the time of this dissertation, the integration of Network

[1] https://fastapi.tiangolo.com/

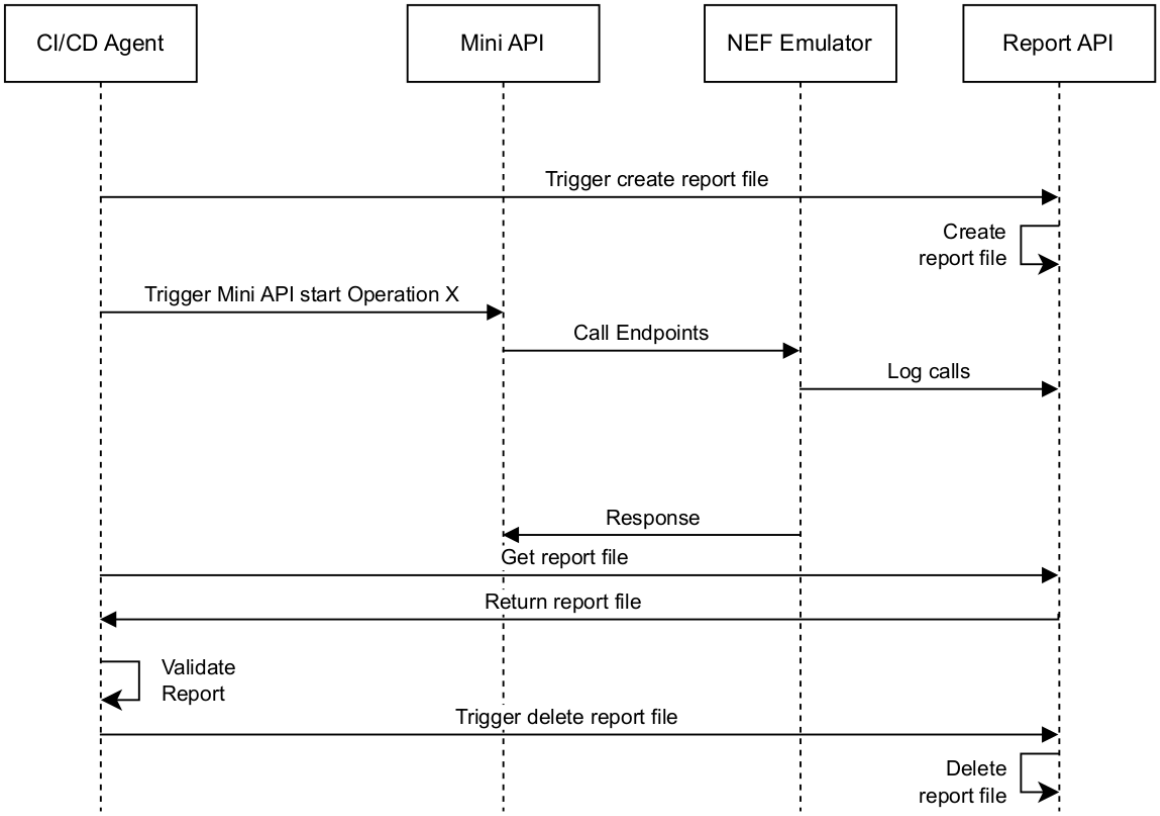

Figure 4.6: Implemented Components' Interaction

Applications with this new concept has not yet been implemented, nor has their interaction with the NEF emulator been established.

Thus, the approach towards the Mini API deviated somewhat from the initial proposal. In this dissertation the Mini API consolidates all the communication logic with the NEF emulator within the start endpoint.

That is all the logic to communicate with the NEF emulator is being done inside the start endpoint of the Mini API due to the lack of maturity of the Network Applications.

The ideal scenario was introduced in Figure 3.2, but with the considerations just made the scenario implemented is described in Figure 4.6.

This API is also nested within a Docker Image and deployed through Docker Compose.

4.4 Validation Tests

After implementing the components proposed for the validation of the Network Applications, the last thing to do is implement the actual tests. As already mentioned in Section 3.1, this dissertation focus is on tests related to the 5G Readiness and Security scopes.

Before presenting the tests developed in the context of this dissertation, it is crucial to establish both the location for their storage and the necessary file specifications as well as to define the structure of these tests.

Once completed the discussion of the storage location and test structure, the tests will be described.

4.4.1 LTR

As mentioned in Section 2.8.4, the LTR is an element of the 5GASP CI/CD Service, that stores all the developed tests and makes them available for the CI/CD Agent to execute them.

The file structure of the LTR is shown in the Code Block 4.1 with three of the tests created in the context of this dissertation:

These tests were developed using the Robot Framework. The tests are composed by: (i) a Python file, which contains code for executing the test and validating the response, (ii) a Robot file, that contains all the Test Cases, (iii) a Markdown [2] file, to describe the test, his inputs and outputs, and (iv) a text file, that contains the requirements needed to run the test.

```
.
|-- nef_login
|   |-- nef_login.py
|   |-- test_nef_login.robot
|   |-- docs.md
|   --- test_nef_login.txt
|-- nef_monitoring_subscription
|   |-- nef_monitoring_subscription.py
|   |-- test_nef_monitoring_subscription.robot
|   |-- docs.md
|   --- requirements.txt
--- nef_ue_location_acquisition
    |-- nef_ue_location_acquisition.py
    |-- test_nef_ue_location_acquisition.robot
    |-- docs.md
    --- requirements.txt
```

Code Block 4.1: File Structure of the LTR.

4.4.2 5G Readiness

The tests defined in this scope aim to verify the ability of a Network Application to communicate and interact with the NEF emulator in order to test the readiness of the Network Application for the 5G Network.

Within this scope, as all the tests use the NEF emulator and consequently the Mini API and the Report API, they have all the same structure. This can be seen in Figure 4.7.

Initially, it is made the call to create the shared file, which is done through a request to the Report API. After this, the start endpoint of the Mini API is invoked, this is when the interactions with the NEF emulator occur and when it logs all the communication done. Finally, the report file generated is retrieved, which is another call to the Report API and then is validated according to the respective test being conducted. Lastly, the request to the Report API to delete the report file generated is done.

[2] https://www.markdownguide.org/

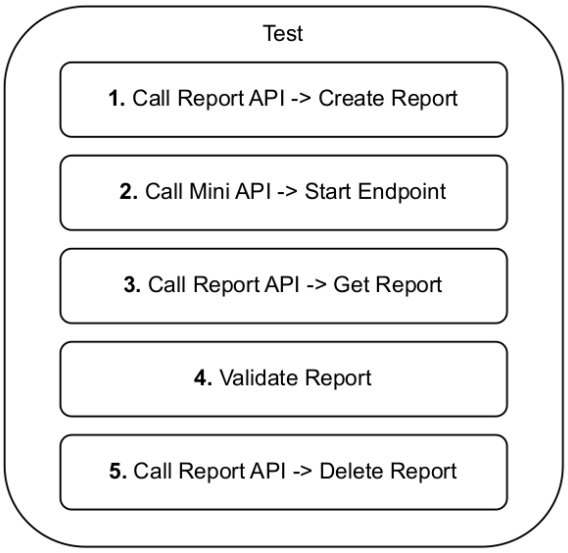

Figure 4.7: Test's Structure

The tests developed in order to test if a Network Application does the correct communication with the NEF emulator are displayed in Table 4.2.

Test	**Description**
Authentication with NEF emulator	Validates the authentication of a Network Application to use the NEF emulator.
Creation of a Monitoring Subscription	Validates if a Network Application is able to subscribe to a Monitoring Event.
Creation of a QoS Subscription	Validates if a Network Application is able to subscribe to a QoS Event.
Acquisition of an UE Location	Validates if a Network Application is able to retrieve an UE location.
Acquisition of an UE Handover Event	Validates if a Network Application is able to retrieve information about a handover event.
Acquisition of an UE Path Losses	Validates if a Network Application is able to retrieve information about path losses.
Acquisition of an UE Serving Cell	Validates if a Network Application is able to retrieve information about the serving cell node
Acquisition of an UE Reference Signal Received Power (RSRP)	Validates if a Network Application is able information about the RSRP.

Table 4.2: Implemented 5G Readiness Tests

The first entry of the Table 4.2, aims to validate if a Network Application can authenticate for access to the NEF emulator. Following this, the second and third tests validate if the Network Application can correctly use the Northbound APIs, the Monitoring Event API and the AsSessionWithQoS API, to subscribe to this services. The remaining five tests featured in the table use the additional endpoints added to the emulator and test whether a Network

Application can get a certain information related to a given UE.

To provide a more complete overview of a test developed in this dissertation, the *nef_monitoring_subscription* test has been selected as an illustrative example. Code Block 6.2, situated in the Appendix, displays the Python file, while Code Block 4.2 presents the Robot file. By examining the two Code Blocks, it becomes clear that the Python file follows the structure defined in Figure 4.7, and, more importantly, it incorporates a function responsible for validating the report against the endpoint and method used and the response generated by the NEF emulator. Finnaly, the validation result produced by this test is verified within the Robot File to determine the success or failure of the test.

```
*** Settings ***
Library       nef_monitoring_subscription.py
Test Timeout  15 minutes

*** Test Cases ***
Testing the creating of a monitoring subscription in NEF
    ${nef_monitoring_subscription_status}=    Test NEF Monitoring Subscription
    ↪  %{nef_monitoring_subscription_report_api_ip}
    ↪  %{nef_monitoring_subscription_report_api_port}
    ↪  %{nef_monitoring_subscription_report_name}
    ↪  %{nef_monitoring_subscription_mini_api_ip}
    ↪  %{nef_monitoring_subscription_mini_api_port}
    ↪  %{nef_monitoring_subscription_nef_ip}
    ↪  %{nef_monitoring_subscription_nef_port}
    ↪  %{nef_monitoring_subscription_nef_user}
    ↪  %{nef_monitoring_subscription_nef_pass}

    IF  '${nef_monitoring_subscription_status[0]}' in ['0']
        Pass Execution  \n${nef_monitoring_subscription_status[1]}
    ELSE IF  '${nef_monitoring_subscription_status[0]}' in ['1', '2', '3', '4']
        Fail  \n${nef_monitoring_subscription_status[1]}
    ELSE
        Fail  \nUnknown Error
    END
```

Code Block 4.2: test_nef_monitoring_subscription.robot

4.4.3 Security

The focus of this dissertation is the 5G Readiness scope, however two tests were developed for the Security scope.

The objective within this scope is to assess whether a Network Application adheres to the security and privacy standards established by 3GPP.

This scope holds significant importance, and within the context of this dissertation, two tests proposed by 5GASP were created. These two tests developed were:

- **Secure Sockets Layer (SSL) Protected APIs Test** - Validates if the Network Application's VNFs offered APIs are protected with SSL. The SSL protocol is used to establish secure and encrypted connections between a client and a server, in other

words ensures the confidentiality and integrity of data transmitted over the internet. Regarding Network Applications, they need to have a SSL certificate to authenticate their identity an to keep their privacy and interactions secure.

- **Openstack Port Security** - Validates if the Network Application's VNFs have the Port Security enabled, when deployed in Openstack. IP forwarding is the ability for an operating system to accept incoming network packets on one interface, recognize that it is not meant for the system itself, but that it should be passed on to another network, and then forwards it accordingly. Concerning the Network Applications they are in their private network, if they access the 5G network and if between the IP forwarding is enabled their traffic could be forwarded to a third network causing a breach on security. This test verifies whether Port Security is enabled when using OpenStack because the Port Security feature in OpenStack blocks IP forwarding.

4.4.4 Generation of Values

A test was developed to validate whether the NEF Emulator can generate values for a specific parameter of a AsSessionWithQoS API asked by a Network Application, taking into account the limitations of this process.

Although the structure of the test is the same as that of the 5G Readiness tests, the function for validating the report is slightly different. Code Block 4.3 illustrates this function.

```
def validate_report(report, params):
    errors = []
    first_value = None
    second_value = None
    subsId = None
    keys = params.split("-")
    for request in report:
        if request["endpoint"] ==
        ↪  f"/nef/api/v1/3gpp-as-session-with-qos/v1/netapp/subscriptions":
            if request["method"] != "POST":
                errors.append("Wrong method used.")
                break
            elif request["nef_response_code"] not in [200, 201 ,409]:
                errors.append(f"Unable to create QoS Subscription due to:
                ↪ {request['nef_response_message']}")
                break
            else:
                msg = request['nef_response_message'].replace("'", "\"")
                json_msg = json.loads(msg)
                subsId = json_msg['link'].split('/')[-1]

                if len(keys) > 1:
                    current_doc = json_msg
                    for key in keys[:-1]:
                        current_doc = current_doc[key]

                    first_value = current_doc[keys[-1]]
```

```
            else:
                first_value = json_msg[params]

for request in report:
    if request["endpoint"] ==
    ↪  f"/nef/api/v1/3gpp-as-session-with-qos/v1/netapp/subscriptions/{subsId}":
        if request["method"] != "GET":
            errors.append("Wrong method used.")
            break
        elif request["nef_response_code"] not in [200, 409]:
            errors.append(f"Unable to get QoS Subscription due to:
            ↪  {request['nef_response_message']}")
            break
        else:
            msg = request['nef_response_message'].replace("'", "\"")
            json_msg = json.loads(msg)

            if len(keys) > 1:
                current_doc = json_msg
                for key in keys[:-1]:
                    current_doc = current_doc[key]

                second_value = current_doc[keys[-1]]
            else:
                second_value = json_msg[params]

if first_value == second_value:
    errors.append("Parameter not updated.")
return errors
```

Code Block 4.3: validate_report function from nef_generate_values.py

What sets this function apart from its counterparts in the 5G Readiness tests is that in this one, two iterations of the report are performed. This is because the first iteration has two objectives. The first objective is to validate the correct creation of a QoS subscription within the NEF emulator. The second objective is to retrieve the *subscriptionId* created by the NEF emulator for that subscription. Additionally, in the first iteration, the initial value of the parameter that is supposed to change is saved. The Network Application, to consume the values generated by the broker, needs to do a *GET* request to retrieve the specific QoS subscription with the previously retrieved *subscriptionId*. This behavior is checked in the second iteration. If the request is made correctly, the value of the desired parameter is saved. In the end, this allows for a comparison of the two saved values. If the first value is different from the last value, it indicates that the generation of values through the NEF was successful.

4.5 Chapter Summary

This Chapter presented the implementation of the components proposed for the validation process and the new tests developed.

Section 4.1 begins to address the NEF Emulator. First, in Section 4.1.1 the Northbound APIs from TS29.122 [55] and TS29.522 [56] that were explored are presented, and a table displayed which of them were added to the emulator is shown. Then in Section 4.1.2 the functionality added to the NEF Emulator for logging the interactions done between the Network Applications and the emulator is explained. Afterwards the endpoints added to support information given by the emulation provided by the NEF emulator are presented in Section 4.1.3. The final point mentioned, regarding the emulator is the Generation of Values, presented in Section 4.1.4, where an explanation of how the implementation was done, a display of the UI provided, the endpoints that allow for the generation of values and a description of how to generate values are presented.

Sections 4.2 and 4.3 present the implementation of the two APIs developed, respectively, the Report API and the Mini API.

Lastly, the validation tests are presented in Section 4.4. Where the 5G Readiness tests, the Security tests and the tests for the generation of values described and explained.

CHAPTER 5

Results

After implementing the proposed solutions, it is now necessary to validate them. Therefore, this Chapter addresses the results obtained through the tests created.

Section 5.1 discusses the 5G Readiness Tests. It begins by describing the methodology used to perform the evaluation of the tests and then presents the results.

Sections 5.2 and 5.3 follow the same structure as Section 5.1 but focus on the Security Tests and the Generation of Values using the NEF Emulator, respectively.

5.1 5G Readiness Tests

The tests developed for the 5G Readiness scope aim to validate the interaction between Network Applications and the NEF emulator.

To support these tests, functionalities were added to the NEF emulator, namely the Report Handler, and two APIs were created: the Report API and the Mini API.

The developed components and tests were integrated within the 5GASP CI/CD pipeline.

However, these tests were not executed in actual Network Applications because they do not yet implement communication with both the NEF Emulator and the Mini API.

5.1.1 Testing Scenario

Although real Network Applications weren't used, it is still possible to use dummy Network Applications that interact with the NEF emulator, as this aligns with the intended purpose of this scope.

Additionally, the tests developed needed some form of validation. However, measuring the tests' execution time, for example, doesn't make much sense as it doesn't provide any value given the nature of the tests.

So, to validate the functionality of these tests, an exhaustive evaluation was done, where several dummy Network Applications were created with or without points of failure.

Two additional points worth mentioning are: (i) the 5G Readiness tests are not mandatory, meaning that a Network Application does not need to be validated against all of them. For

instance, a Network Application could only be interested in subscribing to Monitoring Events, meaning that it doesn't need to subscribe to the QoS, and in the same way, another Network Application that could only need to subscribe to QoS events doesn't need to subscribe to Monitoring Events. It depends on the purpose of the application, and (ii) consequently, since the Network Application is seen as a black box for the tester, the Network Application owners define what the application does in terms of accessing the 5G infrastructure. Therefore, they need to commit to the interactions they make, and validation is based on that. If they commit to communicating with the 5G infrastructure, they must do so correctly.

Thus, to perform the exhaustive evaluation, eight dummy Network Applications were developed, each making multiple requests to the NEF Emulator. Table 5.1 presents the interactions performed by each dummy Network Application.

Network Application	**Interactions**
Dummy Network Application 1	• Authentication on the NEF Emulator. • Create a Monitoring Subscription. • Acquisition of the UEs Location. • Acquisition of the Serving Cell of the UE with Subscription Permanent Identifier (SUPI) 202010000000001. • Create a QoS Subscription.
Dummy Network Application 2	• Authentication on the NEF Emulator. • Create a Monitoring Subscription. • Create a QoS Subscription.
Dummy Network Application 3	• Authentication on the NEF Emulator. • Create a Monitoring Subscription. • Acquisition of the UEs Location. • Acquisition of the RSRPs for the UE with SUPI 202010000000001. • Acquisition of the Path Losses for the UE with SUPI 202010000000001.

Dummy Network Application 4	• Create a Monitoring Subscription. • Acquisition of the Serving Cell of the UE with SUPI 202010000000001. • Acquisition of the Handovers for the UE with SUPI 202010000000002. • Create a QoS Subscription.
Dummy Network Application 5	• Authentication on the NEF Emulator. • Acquisition of the UEs Location. • Acquisition of the Path Losses for the UE with SUPI 202010000000001. • Acquisition of the Serving Cell of the UE with SUPI 202010000000001. • Acquisition of the RSRPs for the UE with SUPI 202010000000001. • Acquisition of the Handovers for the UE with SUPI 202010000000002. • Create a QoS Subscription.
Dummy Network Application 6	• Authentication on the NEF Emulator. • Create a Monitoring Subscription. • Acquisition of the Handovers for the UE with SUPI 202010000000001. • Create a QoS Subscription.
Dummy Network Application 7	• Authentication on the NEF Emulator. • Create a Monitoring Subscription. • Create a QoS Subscription.

Dummy Network Application 8	• Authentication on the NEF Emulator. • Create a Monitoring Subscription. • Acquisition of the Serving Cell of the UE with SUPI 202010000000001. • Acquisition of the Handovers for the UE with SUPI 202010000000003. • Acquisition of the UEs Location. • Acquisition of the Path Losses for the UE with SUPI 202010000000001. • Acquisition of the RSRPs for the UE with SUPI 202010000000001. • Create a QoS Subscription.

Table 5.1: Dummy Network Applications' Communication with NEF Emulator

The Network Applications were developed using several Mini APIs which implies having to onboard several VNFs onto the pipeline.

The VNF and the NS for each Mini API were onboarded onto an OSM Release 12 instance and deployed. The template tags of the NS were used on the Testing Descriptor to define the IP address of the Mini API instance.

The Testing Descriptor should be provided by each Network Application. To show an example of the test cases defined in the Testing Descriptor for Dummy Network Application 2, Code Block 5.1 is presented.

```
testcases:
      - testcase_id: 1
        type: pre-defined
        scope: 5g_readiness
        name: nef_login
        description: Test if the Network Appllication can authenticate withing NEF
        parameters:
          - key: report_api_ip
            value: 10.255.28.183
          - key: report_api_port
            value: 3000
          - key: report_name
            value: report.json
          - key: mini_api_ip
            value: {{deployment_info|miniapi_final-nsd|vnf1|vnf-cp0-ext|ip-address}}
          - key: mini_api_port
            value: 3001
          - key: nef_ip
            value: 10.255.28.183
          - key: nef_port
```

```
      value: 8888
    - key: nef_user
      value: admin1@my-email.com
    - key: nef_pass
      value: pas
    - key: napp_endpoint
      value: /start/2/1

- testcase_id: 2
  type: pre-defined
  scope: 5g_readiness
  name: nef_monitoring_subscription
  description: Test if the Monitoring Subscription of the Network Appllication is right
  parameters:
    - key: report_api_ip
      value: 10.255.28.183
    - key: report_api_port
      value: 3000
    - key: report_name
      value: report.json
    - key: mini_api_ip
      value: {{deployment_info|miniapi_final-nsd|vnf1|vnf-cp0-ext|ip-address}}
    - key: mini_api_port
      value: 3001
    - key: nef_ip
      value: 10.255.28.183
    - key: nef_port
      value: 8888
    - key: nef_user
      value: admin1@my-email.com
    - key: nef_pass
      value: pas
    - key: napp_endpoint
      value: /start/2/2

- testcase_id: 3
  type: pre-defined
  scope: 5g_readiness
  name: nef_qos_subscription
  description: Test if the Network Application is able to do a QoS Subscription
  parameters:
    - key: report_api_ip
      value: 10.255.28.183
    - key: report_api_port
      value: 3000
    - key: report_name
      value: report.json
    - key: mini_api_ip
      value: {{deployment_info|miniapi_final-nsd|vnf1|vnf-cp0-ext|ip-address}}
    - key: mini_api_port
```

```
    value: 3001
  - key: nef_ip
    value: 10.255.28.183
  - key: nef_port
    value: 8888
  - key: nef_user
    value: admin1@my-email.com
  - key: nef_pass
    value: pas
  - key: napp_endpoint
    value: /start/2/3
```

Code Block 5.1: test_nef_monitoring_subscription.robot

5.1.2 Results

Results were collected based on the previously described testing scenario.

The results gathered here will address each individual dummy Network Application. It is important to note that the dummy Network Applications were intentionally designed to fail certain tests at specific known points in order to validate the correctness of the developed tests. In a real-world scenario, it is evident that a Network Application owner develops their application with the belief that it functions properly, and the tests are conducted to verify this belief.

The results displayed in Tables 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8 and 5.9 were obtained through the web interface of the TRVD generated from the executed tests. As illustrative examples, Figures 1 and 2, display the web interface of the TRVD for the Dummy Network Application 5 and the *report.html* created by the Robot Framework for the *nef_handovers* test, respectively, after the validation process was concluded.

Dummy Network Application 1

Interaction with NEF Emulator	**Expected Result**	**Result**
Authentication within the NEF Emulator	Pass	Pass
Create a Monitoring Subscription	Pass	Pass
Acquisition of UEs Location	Pass	Pass
Acquisition of the Serving Cell of the UE with SUPI 202010000000001.	Pass	Pass

Create a QoS Subscription.	Pass	Pass

Table 5.2: Dummy Network Application 1 Results

Dummy Network Application 2

Interaction with NEF Emulator	**Expected Result**	**Result**
Authentication within the NEF Emulator	Fail - Wrong credentials used	Fail - Unable to authenticate due to: Incorrect email or password.
Create a Monitoring Subscription	Fail - Lack of authentication	Fail - Unable to create a monitoring subscription due to: Could not validate credentials
Create a QoS Subscription.	Fail - Lack of authentication	Fail - Unable to create a QoS subscription due to: Could not validate credentials

Table 5.3: Dummy Network Application 2 Results

Dummy Network Application 3

Interaction with NEF Emulator	**Expected Result**	**Result**
Authentication within the NEF Emulator	Fail - Wrong Implementation	Fail - Unable to authenticate due to: Incorrect email or password
Create a Monitoring Subscription	Fail - Lack of authentication due to wrong implementation	Fail - Unable to create a monitoring subscription due to: Could not validate credentials
Acquisition of UEs Location	Fail - Lack of authentication due to wrong implementation	Fail - Unable to get UEs information due to: Could not validate credentials

Acquisition of RSRP for the UE with SUPI 202010000000001	Fail - Lack of authentication due to wrong implementation	Fail - Unable to get RSRP information due to: Could not validate credentials
Acquisition of Path Losses for the UE with SUPI 202010000000001	Fail - Lack of authentication due to wrong implementation	Fail - Unable to get path losses due to: Could not validate credentials

Table 5.4: Dummy Network Application 3 Results

Dummy Network Application 4

Interaction with NEF Emulator	**Expected Result**	**Result**
Create a Monitoring Subscription	Fail - Lack of authentication	Fail - Unable to create a monitoring subscription due to: Not authenticated
Acquisition of the Serving Cell for the UE with SUPI 202010000000001	Fail - Lack of authentication	Fail - Unable to get serving cell information due to: Not authenticated
Acquisition of Handovers for the UE with SUPI 202010000000002	Fail - Lack of authentication	Fail - Unable to get handovers due to: Not authenticated
Create a QoS Subscription.	Fail - Lack of authentication	Fail - Unable to create a QoS subscription due to: Not authenticated

Table 5.5: Dummy Network Application 4 Results

Dummy Network Application 5

Interaction with NEF Emulator	**Expected Result**	**Result**
Authentication within the NEF Emulator	Pass	Pass
Acquisition of UEs Location	Pass	Pass

Acquisition of Path Losses for the UE with SUPI 202010000000001	Fail - Emulation environment needs to be active	Fail - Unable to get path losses due to: The emulation needs to be ongoing
Acquisition of the Serving Cell for the UE with SUPI 202010000000001	Fail - Emulation environment needs to be active	Fail - Unable to get serving cell information due to: The emulation needs to be ongoing
Acquisition of RSRP for the UE with SUPI 202010000000001	Fail - Emulation environment needs to be active	Fail - Unable to get RSRP information due to: The emulation needs to be ongoing
Acquisition of Handovers for the UE with SUPI 202010000000002	Fail - Emulation environment needs to be active	Fail - Unable to get handovers due to: The emulation needs to be ongoing
Create a QoS Subscription.	Pass	Pass

Table 5.6: Dummy Network Application 5 Results

Dummy Network Application 6

Interaction with NEF Emulator	**Expected Result**	**Result**
Authentication within the NEF Emulator	Pass	Pass
Create a Monitoring Subscription	Fail - Wrong payload	Fail - Unable to create a monitoring subscription due to: `['loc': ['body', 'maximumNumberOfReports'], 'msg': 'value is not a valid integer', 'type': 'type_error.integer']`
Acquisition of Handovers for the UE with SUPI 202010000000001	Pass	Pass

Create a QoS Subscription.	Fail - Wrong payload	Fail - Unable to create a QoS subscription due to: `['loc': ['body', 'qosReference'], 'msg': 'value is not a valid integer', 'type': 'type_error.integer']`

Table 5.7: Dummy Network Application 6 Results

Dummy Network Application 7

Interaction with NEF Emulator	**Expected Result**	**Result**
Authentication within the NEF Emulator	Pass	Pass
Create a Monitoring Subscription	Fail - Wrong method used	Fail - Test Failed due to the following errors: Wrong method used.
Create a QoS Subscription.	Pass - Subscription already exists	Pass

Table 5.8: Dummy Network Application 7 Results

Dummy Network Application 8

Interaction with NEF Emulator	**Expected Result**	**Result**
Authentication within the NEF Emulator	Pass	Pass
Create a Monitoring Subscription	Pass	Pass
Acquisition of the Serving Cell for the UE with SUPI 202010000000001	Pass	Pass
Acquisition of Handovers for the UE with SUPI 202010000000003	Pass	Pass

Acquisition of UEs Location	Pass	Pass
Acquisition of Path Losses for the UE with SUPI 202010000000001	Pass	Pass
Acquisition of RSRP for the UE with SUPI 202010000000001	Pass	Pass
Create a QoS Subscription.	Fail- Wrong method used	Fail - Test Failed due to the following errors: Wrong method used.

Table 5.9: Dummy Network Application 8 Results

5.2 Security Tests

5.2.1 Testing Scenario

To test the Security tests developed, three Dummy Network Applications were created, to test the different outcomes of the two security tests.

5.2.2 Results

The results displayed in Tables 5.10, 5.11 and 5.12 were obtained through the web interface of the TRVD generated from the executed tests. The tests validate if the Openstack Port Security is enabled and if the Network Application has a SSL certificate.

Dummy Network Application 9

Test	**Expected Result**	**Result**
Openstack Port Security	Pass	Pass
SSL Audit	Fail - No SSL certificate	Fail - Test Failed due to the following errors: Not SSL protected.

Table 5.10: Dummy Network Application 9 Results

Dummy Network Application 10

Test	**Expected Result**	**Result**

Openstack Port Security	Fail - Openstack Port Security disabled	Fail - Test Failed due to the following errors: Port Security disabled.
SSL Audit	Pass	Pass

Table 5.11: Dummy Network Application 10 Results

Dummy Network Application 11

Test	**Expected Result**	**Result**
Openstack Port Security	Pass	Pass
SSL Audit	Pass	Pass

Table 5.12: Dummy Network Application 11 Results

5.3 Generation of Values

The method for generating values for the AsSessionWithQoS API was added to the NEF emulator in order to provide more real values for the Network Applications to use, if they choose to do so.

5.3.1 Testing Scenario

To test the functionality to Generate Values through the NEF emulator, a twelfth Dummy Network Application was created. The aim of this Network Application is to do a QoS Subscription and then request to generate values for consumption through the NEF Emulator. Given the limitations of the method for generating values, i.e. only works on a parameters at a time, the Network Application has to chose a parameter of their interest to where the values will be attributed. Table 5.13 displays the interactions between the Dummy Network Application and the NEF Emulator.

Network Application	**Interactions**
Dummy Network Application 12	• Authentication on the NEF Emulator. • Create a QoS Subscription. • Generate Values for the *usageThreshold-uplinkVolume* parameter.

Table 5.13: Dummy Network Application 12 Communication with NEF Emulator

5.3.2 Results

Results were collected based on the previously described testing scenario. The results displayed in Table 5.14 were also obtained through the web interface of the TRVD generated from the executed tests.

To clarify the second test on the table, it validated if the Network Application was able to request the NEF Emulator to create a QoS subscription and then if it was capable of doing the requests for the generation of values process for a specific parameter. It validated if the initial value of the parameter was the same as after the process for generating values was carried out.

Dummy Network Application 12

Interaction with NEF Emulator	**Expected Result**	**Result**
Authentication within the NEF Emulator	Pass	Pass
Generate Values with for the *usageThreshold-uplinkVolume* parameter	Pass	Pass

Table 5.14: Dummy Network Application 12 Results

5.4 Chapter Summary

This Chapter begins by addressing the results for the 5G Readiness Tests. In Section 5.1.1 the scenarios in which the tests were used, are presented. Eight Dummy Network Applications were created, to simulate the interaction between a Network Application and the NEF emulator. In these eight Dummy Network Applications, several cases to fail the validation were made on purpose. Section 5.1.2 presents all the tables for the Dummy Network Applications demonstrating the expected results versus the results gathered after the validation process was finished.

After presenting the 5G Readiness Tests, Security tests are addressed in Section 5.2. This Section follows the same structure as the previous one. Section 5.2.1 addresses the scenario where the tests were performed and after 5.2.2 presents the results obtained regarding this scope of tests.

Lastly, the Generation of Values is addressed in Section 5.3, where a twelfth Dummy Network Application was created to simulate the interactions needed to request the NEF emulator to produce values for a AsSessionWithQoS subscription. As before, the testing scenario is first addressed in Section 5.3.1 and then the results obtained after executing the tests are presented in Section 5.3.2.

CHAPTER 6

Conclusions

6.1 Conclusions

With the transition into the world of NFV, Network Operators face the challenge of making sure VNFs perform correctly. The current approach to validate VNFs leans heavily on automation. Projects like VITAL-5G, EVOLVED-5G, 5GMediaHUB, and 5GASP, among others not mentioned here, prove that they are striving for full automation in the Network Applications validation process.

The work presented in this dissertation is aligned with the 5GASP's goals centers around two primary goals: (i) the development of tests designed to validate Network Applications and (ii) the extension of the NEF Emulator to enable the creation of tests that utilize it.

Regarding the second objective, several Northbound APIs that follow the TS29.122 [55] and TS29.522 [56] standards were added to the NEF emulator providing more capabilities exposed to external applications. However, the 5GASP's Network Applications ended not using them so tests around them weren't developed either.

To reach the development of tests, the flow of how they would run had to be defined, which lead to the creating of a Report API with the objective to manage the file shared between this API and the NEF Emulator, where the emulator registers the interactions made between the NEF emulator and the Network Applications. This API was very simple and worked for the desired outcome.

The Mini API, was also a outcome of this dissertation and it is being heavily used in the 5GASP project. It was created with the purpose of controlling the flow of the interactions made between the Network Applications and the NEF Emulator. Testers can't access the code inside Network Applications, so it was crucial to manage when these interactions happened, especially when some CI/CD Service components might not be available. The Mini API ended to not communicate with an actual Network Application but did successfully trigger the interactions that were implemented on it.

Regarding the development of tests, 5G Readiness tests were the main focus of this work. These tests were built around two existing APIs, the MonitoringEvent API and

AsSessionWithQoS API, since Network Application developers had more interest in these capabilities. To evaluate the tests and see if they could verify the correct behavior of a Network Application, eight Dummy Network Applications were developed. In these dummy applications, several tests were made to fail on purpose, to validate if the test could catch the errors. For instance, the dummy Network Applications could not be able to authenticate with the NEF Emulator, or could be requesting for the creation of a subscription providing wrong payloads, among others. All the results obtained for the Dummy Network Applications were the expected.

The work in this dissertation also explored the possibility to generate more close to real values for the Network Applications to consume through the NEF emulator. This functionality, although limited, was also a successful addition. Also serves the purpose to demonstrate that a more deep exploration of this functionality is possible and feasible.

The work developed in this dissertation was an important outcome for evaluating if Network Applications are 5G ready. With the validation proposed on this dissertation, we are able to know if Network Applications are or not prepared to communicate with a general NEF.

6.2 Future Work

In terms of implementing additional Northbound APIs, it's important to note that not all of these have been integrated. This means there is an opportunity to expand the functionality of the NEF Emulator by incorporating these remaining APIs. It's worth mentioning that due to time constraints, there was not the opportunity to conduct tests utilizing these new APIs.

The inclusion of more Northbound APIs brings about the potential for enhancing the emulator's capability in terms of emulation. It offers the possibility to add new features that interact with the new integrated APIs, thereby increasing the utility and realism of the emulator.

Regarding the feature for generating more realistic values for Network Applications, there is room for improvement in terms of flexibility. The process can be made less restrictive in two key aspects. First, it can be designed to be more versatile, allowing a broader range of APIs to benefit from this data generation process. Second, it can be made less restrictive in terms of the number and types of parameters for which values are generated, aiming for a more generic approach applicable to all Northbound APIs.

Expanding the range of tests is another point for further development. Diversifying the testing suite to cover the Northbound APIs that have been added but not yet utilized is important. The introduction of additional APIs requires the development of tests that effectively utilize them and requires the actualization of the report handler to support them.

Furthermore, it is essential to highlight that, at the time of writing this dissertation, real Network Applications were not prepared to use the validation the process described in this work, leading to their non-utilization. In the future, a significant focus is to conduct the testing using real Network Applications with the proposed and implemented solution, which will be done in the 5GASP project.

References

[1] C. Bouras, P. Ntarzanos, and A. Papazois, "Cost modeling for SDN/NFV based mobile 5G networks," in *2016 8th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 2016, pp. 56–61. DOI: 10.1109/ICUMT.2016.7765232.

[2] R. Direito, D. Gomes, and R. L. Aguiar, "Towards a Fully Automated System for Testing and Validating NetApps," in *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*, 2022, pp. 231–233. DOI: 10.1109/NetSoft54395.2022.9844037.

[3] M. Chiosi and et all, "White paper: Network Functions Virtualisation - An Introduction, Benefits, Enablers, Challenges & Call for Action," Tech. Rep. 1, Oct. 2012.

[4] *5GASP*, Accessed 2023. [Online]. Available: https://5g-ppp.eu/5gasp/.

[5] S. Rommer, P. Hedman, M. Olsson, L. Fried, S. Sultana, and C. Mulligan, *5G Core Networks: Powering Digitalization*. Elsevier/Academic Press, 2020, ISBN: 978-0-08-103009-7. [Online]. Available: https://www.oreilly.com/library/view/5g-mobile-core/9781484264737/.

[6] R. Chayapathi, S. F. Hassan, and P. Shah, *Network Functions Virtualization (NFV) with a Touch of SDN*, [Online; accessed 2023], 2016. [Online]. Available: https://ptgmedia.pearsoncmg.com/images/9780134463056/samplepages/9780134463056.pdf.

[7] ETSI ISG NFV, "ETSI GS NFV 002 V1.1.1: Network Function Virtualisation (NFV): Architectural Framework," Tech. Rep. 1, Oct. 2013.

[8] M. Abu-Lebdeh, D. Naboulsi, R. Glitho, and C. W. Tchouati, "NFV orchestrator placement for geo-distributed systems," in *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*, 2017, pp. 1–5. DOI: 10.1109/NCA.2017.8171391.

[9] *Network Functions Virtualisation (NFV)*, Accessed 2023. [Online]. Available: https://www.etsi.org/technologies/nfv.

[10] ETSI ISG NFV, "ETSI GS NFV-MAN 001 - V1.1.1 - Network Functions Virtualisation (NFV); Management and Orchestration," Tech. Rep. 1, Dec. 2014.

[11] L. Networks, "White Paper: Orchestration," 2018, [Online; accessed 2022]. [Online]. Available: https://www.onap.org/wp-content/uploads/sites/20/2019/07/35752846-0-Lumina-Orchestration-2.pdf.

[12] *ONAP Documentation - Architecture*, Accessed 2022. [Online]. Available: https://docs.onap.org/en/latest/platform/architecture/index.html.

[13] *Open Source Mano*, Accessed 2023, Available: https://osm.etsi.org/. [Online]. Available: https://osm.etsi.org/.

[14] *ETSI GS NFV-SOL 005, Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point*, Mar. 2023. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/005/04.04.01_60/gs_NFV-SOL005v040401p.pdf.

[15] *ETSI GS NFV-SOL 006, Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; NFV descriptors based on YANG Specification*, Mar. 2023. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/006/03.07.01_60/gs_NFV-SOL006v030701p.pdf.

[16] ETSI, "OSM Release FOUR - Release Notes," Tech. Rep. 1, May 2018.

[17] OSM - Technical Steering Committee, "OSM Release NINE - Release Notes," Tech. Rep. 1, Dec. 2020.

[18] OSM - Technical Steering Committee, "OSM Release TEN - Release Notes," Tech. Rep. 1, Jun. 2021.

[19] ETSI, "Experience wih NFV Architecture, Interfaces, and Information Models," Tech. Rep. 1, May 2018.

[20] L. Mamushiane, A. A. Lysko, T. Mukute, J. Mwangama, and Z. D. Toit, "Overview of 9 Open-Source Resource Orchestrating ETSI MANO Compliant Implementations: A Brief Survey," in *2019 IEEE 2nd Wireless Africa Conference (WAC)*, 2019, pp. 1–7. DOI: 10.1109/AFRICA.2019.8843421.

[21] ETSI, "OSM Scope, Functionality, Operation and Integration Guidelines," Tech. Rep. 1, Dec. 2019.

[22] A. Hermosilla, J. Gallego-Madrid, P. Martinez-Julia, *et al.*, "Deployment of 5g network applications over multidomain and dynamic platforms," in *2022 IEEE Future Networks World Forum (FNWF)*, 2022, pp. 276–281. DOI: 10.1109/FNWF55208.2022.00055.

[23] T. Xirofotos, "NetApp: Opening up 5G and beyond networks," Tech. Rep. 1, Sep. 2022.

[24] European Commission, *Horizon 2020 - Work Programme 2018-2020 - Information and Communication Technologies*, Oct. 2020. [Online]. Available: https://ec.europa.eu/research/participants/data/ref/h2020/wp/2018-2020/main/h2020-wp1820-leit-ict_en.pdf.

[25] *ETSI TS 123 501 V16.6.0, 5G; System System architecture for the 5G System (5GS)*, Oct. 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.06.00_60/ts_123501v160600p.pdf.

[26] *5G; System architecture for the 5G System (5GS) (3GPP TS 23.501 version 16.18.0 Release 16)*, [Online; accessed 2023], 2023. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.18.00_60/ts_123501v161800p.pdf.

[27] W. Stallings, "Chapter 3: Overview of 5G Use Cases and Architecture," in *5G Wireless: A comprehensive introduction*. Addison Wesley Professional, 2021, pp. 87–93, ISBN: 9780136767206. [Online]. Available: https://www.oreilly.com/library/view/5g-wireless-a/9780136767206/ (visited on 12/12/2022).

[28] R. S. Shetty, *5G Mobile Core Network: Design, Deployment, Automation, and Testing Strategies*. Apress, 2021, pp. 1–372. DOI: https://doi.org/10.1007/978-1-4842-6473-7. [Online]. Available: https://www.oreilly.com/library/view/5g-mobile-core/9781484264737/ (visited on 12/12/2022).

[29] D. Tsolkas and H. Koumaras, "On the development and provisioning of vertical applications in the beyond 5g era," *IEEE Networking Letters*, vol. 4, no. 1, pp. 43–47, 2022. DOI: 10.1109/LNET.2022.3142088.

[30] D. Fragkos, G. Makropoulos, A. Gogos, H. Koumaras, and A. Kaloxylos, "Nefsim: An open experimentation framework utilizing 3gpp's exposure services," in *2022 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, 2022, pp. 303–308. DOI: 10.1109/EuCNC/6GSummit54941.2022.9815829.

[31] M. Krief, *Learning DevOps - Second Edition*. Packt Publishing, 2022. [Online]. Available: https://oreilly.com/library/view/learning-devops/9781801818964/.

[32] L. Z. Len Bass Ingo Weber, *DevOps: A Software Architect's Perspective*. Addison-Wesley Professional, 2015. [Online]. Available: https://oreilly.com/library/view/devops-a-software/9780134049885/.

[33] S. Rossel, *Continuous Integration, Delivery, and Deployment*. Packt Publishing, 2017. [Online]. Available: https://oreilly.com/library/view/continuous-integration-delivery/9781787286610/.

[34] H. van Merode, *Continuous Integration (CI) and Continuous Delivery (CD): A Practical Guide to Designing and Developing Pipelines*. Apress, 2023. [Online]. Available: https://oreilly.com/library/view/continuous-integration-ci/9781484292280/.

[35] *The Convergence of NFV and DevOps for Accelerated Innovation*, Accessed: 19-09-2023. [Online]. Available: https://dzone.com/articles/convergence-of-nfv-amp-devops-for-accelerating-inn.

[36] *SDN/NFV DevOps: Release Automation for Network Operators*, Accessed: 19-09-2023. [Online]. Available: https://devops.com/sdn-nfv-devops-release-automation-for-network-operators/.

[37] *ETSI GR NFV-TST 006, Network Functions Virtualisation (NFV); Testing; Report on CICD and DevOps*, Dec. 2022. [Online]. Available: https://www.etsi.org/deliver/etsi_gr/NFV-TST/001_099/006/01.02.01_60/gr_NFV-TST006v010201p.pdf.

[38] *Best Test Automation Frameworks*, Accessed 2023, Available: https://www.browserstack.com/guide/best-test-automation-frameworks. [Online]. Available: https://www.browserstack.com/guide/best-test-automation-frameworks.

[39] *Robot Framework*, Accessed 2023. [Online]. Available: https://robotframework.org/.

[40] *VITAL-5G*, Accessed 2023. [Online]. Available: https://www.vital5g.eu/.

[41] *Target Use Cases - VITAL-5G*, Accessed 2023. [Online]. Available: https://www.vital5g.eu/target-use-cases/.

[42] VITAL-5G, "D1.4 Testing and validation methodology," Tech. Rep., Dec. 2021.

[43] VITAL-5G, "D2.1 Initial Network Application blueprints and Open Repository design," Tech. Rep., Apr. 2023.

[44] VITAL-5G, "D1.2 System Specifications and Architecture," Tech. Rep., Dec. 2021.

[45] VITAL-5G, "D1.1 Report on use case requirements," Tech. Rep., Jun. 2021.

[46] K. Trichias, G. Landi, E. Seder, *et al.*, "VITAL-5G: Innovative Network Applications (NetApps) Support over 5G Connectivity for the Transport & Logistics Vertical," in *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, 2021, pp. 437–442. DOI: 10.1109/EuCNC/6GSummit51104.2021.9482437.

[47] VITAL-5G, "D4.1 Trial planning and experimentation methodology," Tech. Rep., Jun. 2022.

[48] *EVOLVED-5G*, Accessed 2023. [Online]. Available: https://evolved-5g.eu/.

[49] *Objectives - EVOLVED-5G*, Accessed 2023. [Online]. Available: https://evolved-5g.eu/objectives/.

[50] EVOLVED-5G, "D2.3 Overall framework for NetApp development and evaluation," Tech. Rep., Oct. 2022.

[51] EVOLVED-5G, "D2.2 Design of NetApps development and evaluation environments," Tech. Rep., Oct. 2021.

[52] EVOLVED-5G, "D3.2 NetApp Certification Tools and Marketplace development (intermediate)," Tech. Rep., Jun. 2022.

[53] *LTE; 5G; Common API Framework for 3GPP Northbound APIs (3GPP TS 23.222 version 17.7.0 Release 17)*, [Online; accessed 2023], 2022. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123200_123299/123222/17.07.00_60/ts_123222v170700p.pdf.

[54] EVOLVED-5G, "D3.1 Implementations and integrations towards EVOLVED-5G framework realisation (intermediate)," Tech. Rep., Dec. 2021.

[55] *Universal Mobile Telecommunications System (UMTS); LTE; 5G; T8 reference point for Northbound APIs (3GPP TS 29.122 version 17.10.0 Release 17)*, [Online; accessed 2023], 2023. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/129100_129199/129122/17.10.00_60/ts_129122v171000p.pdf.

[56] *5G; 5G System; Network Exposure Function Northbound APIs; Stage 3 (3GPP TS 29.522 version 17.10.0 Release 17)*, [Online; accessed 2023], 2023. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/129500_129599/129522/17.10.00_60/ts_129522v171000p.pdf.

[57] EVOLVED-5G, "D4.1 5G Exposure Capabilities for Vertical Applications (Intermediate)," Tech. Rep., Feb. 2021.

[58] *Home - 5GMediaHUB*, Accessed 2023. [Online]. Available: https://www.5gmediahub.eu/.

[59] 5GMediaHUB, "D1.8 Architecture Design & Technical Specifications - Final," Tech. Rep., Feb. 2023.

[60] 5GMediaHUB, "D1.2 Use Case Analysis and Target KPIs - Final," Tech. Rep., Jan. 2023.

[61] 5GMediaHUB, "D1.6 DevOps Implementation and Testing Methodology and Benchmarking of Results - Final," Tech. Rep., Feb. 2023.

[62] A. Siokis, K. Ramantas, G. Margetis, *et al.*, "5G experimentation environment for third party media services: the 5GMediaHUB project," in *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, 2021, pp. 1–6. DOI: 10.1109/MeditCom49071.2021.9647663.

[63] *Use Cases - 5GMediaHUB*, Accessed 2023. [Online]. Available: https://www.5gmediahub.eu/use-cases/.

[64] K. Trantzas, C. Tranoris, S. Denazis, *et al.*, "An automated CI/CD process for testing and deployment of Network Applications over 5G infrastructure," in *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, 2021, pp. 156–161. DOI: 10.1109/MeditCom49071.2021.9647628.

[65] 5GGASP, "D5.1 Initial Report on Test Plan Creation and Testing Methodologies," Tech. Rep. 1, Jan. 2023.

[66] 5GGASP, "D5.3 Development of testing tools and framework for the Automotive and PPDR Verticals," Tech. Rep. 1, Apr. 2022.

[67] 5GGASP, "D2.3 : The 5GASP Revised Reference Architecture and Community Components," Tech. Rep. 1, Jul. 2023.

[68] 5GGASP, "D5.4 Initial Report for the Network Application certification testing," Tech. Rep. 1, Mar. 2023.

Appendix

Report Handler Function

```
class ReportLogging(APIRoute):
    def get_route_handler(self) -> Callable:
        original_route_handler = super().get_route_handler()

        async def custom_route_handler(request: Request) -> Response:
            try:
                request_body = {}
                try:
                    request_body = await request.json()
                except JSONDecodeError:
                    pass

                response = await original_route_handler(request)

                extra_fields = {
                    'endpoint': request.url.path,
                    'method': request.method,
                    'request_body': request_body,
                    **self.get_query_params(request.query_params),
                    'nef_response_code': response.status_code,
                    'nef_response_message':
                    ↪  response.body.decode(response.charset).replace('"', "'"),
                }

                self.update_log_file(extra_fields)

                return response
            except (RequestValidationError, HTTPException) as exc:
                status_code = 422 if isinstance(exc, RequestValidationError) else
                ↪  exc.status_code

                extra_fields = {
                    'endpoint': request.url.path,
                    'method': request.method,
                    'request_body': exc.body if isinstance(exc, RequestValidationError) else
                    ↪  request_body,
                    **self.get_query_params(request.query_params),
                    'nef_response_code': status_code,
```

```
                'nef_response_message': exc.detail if isinstance(exc, HTTPException) else
                ↪  exc.errors(),
            }

            self.update_log_file(extra_fields)

            raise HTTPException(status_code=status_code, detail=exc.detail if
            ↪  isinstance(exc, HTTPException) else exc.errors())

    return custom_route_handler

def read_log_file(self):
    with open(settings.REPORT_PATH) as fp:
        return json.load(fp)

def write_log_file(self, listObj):
    with open(settings.REPORT_PATH, 'w') as json_file:
        json.dump(listObj, json_file, indent=4, separators=(',', ': '))

def update_log_file(self, extra_fields):
    global logs_count

    logs_count += 1

    log_entry = {
        'id': logs_count,
        **extra_fields
    }

    listObj = self.read_log_file()
    listObj.append(log_entry)
    self.write_log_file(listObj)

def get_query_params(self, request_query_params):
    query_params = {
        'scsAsId': None,
        'afId': None,
        'subscriptionId': None,
        'transactionId': None,
        'configurationId': None,
        'provisioningId': None,
        'setId': None,
    }

    for param_name in query_params:
        if param_name in request_query_params:
            query_params[param_name] = request_query_params[param_name]

    return query_params
```

Code Block 6.1: Report Handler

Create Monitoring Subscription Test

```
import requests
import json

def validate_report(report):
    errors = []
    for request in report:
        if request["endpoint"] ==
        ↪  "/nef/api/v1/3gpp-monitoring-event/v1/netapp/subscriptions":
            if request["method"] != "POST":
                errors.append("Wrong method used.")
                break
            elif request["nef_response_code"] not in [200, 409]:
                errors.append(f"Unable to create a monitoring subscription due to:
                ↪  {request['nef_response_message']}")
                break
    return errors

def test_nef_monitoring_subscription(report_api_ip, report_api_port, report_name,
↪  mini_api_ip, mini_api_port, nef_ip, nef_port, nef_user, nef_pass):

    try:
        report_api_url = f"http://{report_api_ip}:{report_api_port}" + "/report/"

        # 1. Delete any existing report
        response = requests.delete(report_api_url)

        if not (response.status_code == 200 or response.status_code == 404):
            print(f"Error deleting the report")
            raise Exception(f"Error deleting the report")

        # 2. Create Report
        response = requests.post(report_api_url)

        if response.status_code == 409:
            print(f"Report cound't be created")
            raise Exception(f"Report cound't be created")

        # 3. Call the MiniAPI to run the code
        mini_api_url = f"http://{mini_api_ip}:{mini_api_port}" + "/start/"

        data = {
            "configId": 1,
            "duration": 10,
            "nef_ip": nef_ip,
            "nef_port": nef_port,
            "nef_username": nef_user,
            "nef_pass": nef_pass
        }
```

```
    response = requests.post(mini_api_url, params=data)

    if response.status_code != 200:
        print(f"Error while calling the MiniAPI")
        raise Exception(f"Error while calling the MiniAPI")

    # 4. Get Report
    response = requests.get(report_api_url)

    if response.status_code == 404:
        print(f"Report does not exist")
        raise Exception(f"Report does not exist")

    report = response.json()

    with open(report_name, 'w', encoding='utf-8') as f:
        json.dump(report, f, ensure_ascii=False, indent=4)
except Exception as e:
        print(f"An error occured. Exception {e}")
        return 2, f"An error occured. Exception {e}"

# 5. Validate the Report
errors = validate_report(report)

if len(errors) != 0:
    errors_str = '\n\t- '.join(errors)
    print(f"Test Failed due to the following errors: {errors_str}")
    return 1, f"Test Failed due to the following errors: {errors_str}"
else:
    print("Successful test.")
    return 0, f"Successful test."
```

Code Block 6.2: nef_monitoring_subscription.py

TRVD's Web Interface

20	nef_login	2023-10-19 11:57:29	2023-10-19 11:57:30	Passed	Test if the Network Appllication can authenticate withing NEF	Test Log	Test Report
21	nef_ue_location_acquisition	2023-10-19 11:57:39	2023-10-19 11:57:40	Passed	Test if the Network Application is able to retrieve the UE location	Test Log	Test Report
22	nef_path_losses	2023-10-19 11:57:49	2023-10-19 11:57:50	Failed	Test if the Network Application is able to retrieve the path losses	Test Log	Test Report
23	nef_serving_cell	2023-10-19 11:57:59	2023-10-19 11:57:59	Failed	Test if the Network Application is able to retrieve the service cell information	Test Log	Test Report
24	nef_rsrp	2023-10-19 11:58:09	2023-10-19 11:58:09	Failed	Test if the Network Application is able to retrieve the RSRP	Test Log	Test Report
25	nef_handovers	2023-10-19 11:58:19	2023-10-19 11:58:19	Failed	Test if the Network Application is able to retrieve handovers	Test Log	Test Report
26	nef_qos_subscription	2023-10-19 11:58:29	2023-10-19 11:58:29	Passed	Test if the Network Application is able to do a QoS Subscription	Test Log	Test Report

Figure 1: TRVD's Web Interface Portraiting an Unsuccessful Validation Process

REPORT

Nef Handovers-Test-Id-25 Log

Generated
20231019 12:58:19 UTC+01:00
4 days 2 hours ago

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	1	0	1	0	00:00:00	

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
No Tags						

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Nef Handovers-Test-Id-25	1	0	1	0	00:00:01	
Nef Handovers-Test-Id-25. Test Nef Handovers	1	0	1	0	00:00:01	

Test Execution Log

SUITE **Nef Handovers-Test-Id-25** 00:00:00.555

Full Name: Nef Handovers-Test-Id-25
Source: /var/lib/jenkins/test_repository/YoGoKo-CITS-NetApp-CITS-66/pre-defined-tests/nef_handovers-test-id-25
Start / End / Elapsed: 20231019 12:58:19.367 / 20231019 12:58:19.922 / 00:00:00.555
Status: 1 test total, 0 passed, 1 failed, 0 skipped

SUITE **Test Nef Handovers** 00:00:00.523

Full Name: Nef Handovers-Test-Id-25.Test Nef Handovers
Source: /var/lib/jenkins/test_repository/YoGoKo-CITS-NetApp-CITS-66/pre-defined-tests/nef_handovers-test-id-25/test_nef_handovers.robot
Start / End / Elapsed: 20231019 12:58:19.398 / 20231019 12:58:19.921 / 00:00:00.523
Status: 1 test total, 0 passed, 1 failed, 0 skipped

TEST **Testing the Handovers Information** 00:00:00.452

Full Name: Nef Handovers-Test-Id-25.Test Nef Handovers.Testing the Handovers Information
Timeout: 15 minutes
Start / End / Elapsed: 20231019 12:58:19.468 / 20231019 12:58:19.920 / 00:00:00.452
Status: FAIL
Message:
Test Failed due to the following errors: Unable to get handovers due to: The emulation needs to be ongoing

KEYWORD ${nef_handovers_status} = nef_handovers. **Test Nef Handovers** %{nef_handovers_report_api_ip}, %{nef_handovers_report_api_port}, %{nef_handovers_report_name}, %{nef_handovers_mini_api_ip}, %{nef_handovers_mini_api_port}, %{nef_handovers_nef_ip}, %{nef_handovers_nef_port}, %{nef_handovers_nef_user}, %{nef_handovers_nef_pass}, %{nef_handovers_nef_supi}, %{nef_handovers_napp_endpoint} 00:00:00.447

IF **'${nef_handovers_status[0]}' in ['0']** 00:00:00.001

ELSE IF **'${nef_handovers_status[0]}' in ['1', '2', '3', '4']** 00:00:00.002

Start / End / Elapsed: 20231019 12:58:19.917 / 20231019 12:58:19.919 / 00:00:00.002

KEYWORD Builtin. **Fail** \n${nef_handovers_status[1]} 00:00:00.001

Documentation: Fails the test with the given message and optionally alters its tags.
Start / End / Elapsed: 20231019 12:58:19.918 / 20231019 12:58:19.919 / 00:00:00.001
12:58:19.919 FAIL
Test Failed due to the following errors: Unable to get handovers due to: The emulation needs to be ongoing

ELSE 00:00:00.001

Figure 2: *report.html* Robot File for the Unsuccessful Execution of the NEF Handovers Test