H2020 5GASP Project

Grant No. 101016448

D4.1 Initial Methodology for Designing and Developing NetApps

Abstract

The design and development of NetApps is a key phase in the 5GASP workflow, as it is where the majority of the effort of SMEs is focused. Defining a good methodology to support the design and development of NetApps will help make the 5G ecosystem more accessible to SMEs. It will be a founding pillar of the envisioned 5GASP framework. Supposing the design and development of the NetApps are well-defined, the path through the validation and certification of a NetApp can be significantly shortened towards the final goal of publishing it in the NetApp Store. This document details the initial methodology and the onboarding procedure, which is the connection point between the design and development and the 5GASP portal. The defined processes are based on the first proposed architecture for the 5GASP project and the conducted studies of similar approaches in past projects such as 5GTANGO or 5GEVE.

Document properties

Document number	D4.1
Document title	Initial Methodology for Designing and Developing NetApps
Document responsible	Antonio Skarmeta
Document editor	Jorge Gallego-Madrid, Ana Hermosilla
Editorial team	OdinS
Target dissemination level	PU
Status of the document	Final
Version	1

Document history

Revision	Date	Issued by	Description
0.1	11/06/2021	OdinS	Initial draft
0.2	23/06/2021	UnivBRIS, VMware, YoGoKo	Internal Review
0.3	29/06/2021	OdinS	Submission version

List of Authors

Company	Name	Contribution
OdinS	Jorge Gallego-Madrid Ana Hermosilla Antonio Skarmeta	Abstract, Introduction, NetApp Development, NetApp Management, NetApp Design, Development and Onboarding example
Lamda Networks	Leonidas Lymberopoulos	NetApp Design and NetApps as CNFs
ORO	Elena-Madalina Oproiu Oana Badita Ioan Constantin Andreea Bonea Marius Iordache	Architecture, Integration in 5GASP Architecture
UoP	Christos Tranoris Kostis Trantzas	Onboarding configuration, Ecosystem Workflow, NetApp Management, Network Slice Descriptors
BLB	Yevgeniya Sulema	NetApp Testing and Validation Lifecycle
DriveU	Eli Shapira	NetApp Lifecycle

Disclaimer

This document has been produced in the context of the 5GASP Project. The research leading to these results has received funding from the European Community's H2020 Programme under grant agreement number 101016448.

Contents

ABSTRACT **1**

DOCUMENT PROPERTIES **2**

DOCUMENT HISTORY **2**

LIST OF AUTHORS **2**

DISCLAIMER **2**

CONTENTS **3**

LIST OF FIGURES **4**

LIST OF TABLES **4**

LIST OF ACRONYMS **4**

DEFINITIONS **5**

1. INTRODUCTION **6**

1.1. OBJECTIVES OF THIS DOCUMENT 6
1.2. APPROACH AND METHODOLOGY 6
1.3. DOCUMENT STRUCTURE 7

2. NETAPP ONBOARDING OVERVIEW **8**

2.1. ARCHITECTURE 8
2.1.1. Integration in 5GASP architecture *8*
2.1.2. Onboarding configuration *11*
2.2. ECOSYSTEM WORKFLOW 12

3. NETAPP DESIGN AND DEVELOPMENT METHODOLOGY **15**

3.1. LIFECYCLE 15
3.1.1. Design *15*
3.1.2. Development *16*
3.1.3. Testing *16*
3.1.4. Validation *17*
3.2. DESIGN 18
3.3. DEVELOPMENT 19

4. ONBOARDING APPROACH **21**

4.1. NETAPP MANAGEMENT 21
4.1.1. NetApps categories *21*
4.1.2. NetApps descriptor *23*
4.2. NETAPP DESIGN, DEVELOPMENT AND ONBOARDING EXAMPLE 27
4.2.1. vOBU NetApp design *27*
4.2.2. vOBU NetApp development *29*
4.2.3. vOBU NetApp onboarding *33*

5. CONCLUSIONS **34**

BIBLIOGRAPHY **35**

List of Figures

FIGURE 2.1. 5GASP APPROACH ON DEVOPS EXPERIMENTATION AND CERTIFICATION READINESS LIFECYCLE 8
FIGURE 2.2. 5GASP ARCHITECTURE SIMPLIFIED OVERVIEW 9
FIGURE 2.3. 5GASP DEVOPS ARCHITECTURE 10
FIGURE 2.4. 5GASP DEPLOYMENT AND CI/CD PIPELINE 11
FIGURE 2.5. UNIFIED EXPERIMENTAL MODEL 12
FIGURE 2.6. ONBOARDING AND CONNECTION TO CI/CD SERVICE 13
FIGURE 3.1. 5GASP LIFECYCLE 15
FIGURE 3.2. PHASES OF THE DESIGN OF A NETAPP IN THE CONTEXT OF 5GASP 15
FIGURE 3.3. PHASES OF THE DEVELOPMENT OF A NETAPP IN THE CONTEXT OF 5GASP 16
FIGURE 3.4. TESTING STAGES OF A NETAPP IN THE CONTEXT OF 5GASP 16
FIGURE 4.1. PROPOSED 3GPP ARCHITECTURE FOR CLOUD READY NETWORK FUNCTIONS 23
FIGURE 4.2. GST NETWORK SLICE LIFECYCLE 25
FIGURE 4.3. VIRTUAL ON-BOARD UNIT (VOBU) NETAPP ARCHITECTURE 27

List of Tables

TABLE 4.1. NETAPP4: MULTI-DOMAIN MIGRATION NETAPP'S NEST 26
TABLE 4.3. VOBU NETAPP KPIS 29
TABLE 4.4. VOBU NETAPP NSD 30
TABLE 4.5. VOBU NETAPP VOBU VNFD 31
TABLE 4.6. VOBU NETAPP'S NEST 31
TABLE 4.7. ROBOT TEST DEFINITION 32
TABLE 4.8. PYTHON TEST SCRIPT 32

List of Acronyms

5GASP	5G Application & Services experimentation and certification Platform
API	Application Programming Interface
BW	Bandwidth
CI/CD	Continuous Integration and Continuous Deployment
CNF	Cloud-native Network Function
DevOps	Development and Operations
DSP	Digital Services Provider
GST	Generic Network Slice Template
KPI	Key Performance Indicator
MNO	Mobile network operator
NEST	Network Slice Type
NetApps	Network Applications
NFV	Network Function Virtualization
NFVI	Network Functions Virtualization Infrastructure
NFVO	NFV Orchestrator
NS	Network Services
NSD	Network Service Descriptor
NSP	Network Service Provider

OSM	Open-source MANO
PLR	Packet Loss Ratio
PNFD	Physical Network Function Descriptor
RT	Real-Time
SMEs	Small and Medium-sized Enterprises
V&V	Validation and Verification
V2C/C2V	Vehicle-to-Cloud / Cloud-to-Vehicle
VM	Virtual Machine
VM	Virtual Machine
VNF	Virtual Network Function
VNFD	Virtual Network Function Descriptor
5GASP	5G Application & Services experimentation and certification Platform

Definitions

This document contains specific terms to identify elements and functions that are considered to be mandatory, strongly recommended or optional. These terms have been adopted for use similar to that in IETF RFC2119 and have the following definitions:

- **MUST** This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
- **MUST NOT** This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.
- **SHOULD** This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- **SHOULD NOT** This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- **MAY** This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides).

1. Introduction

1.1. Objectives of this document

The main objective of this document is to present the initial proposal of the unified methodology approach to design, develop and onboard NetApps within the scope of the 5GASP project, letting them use vertical-specific 5G facilities by following standard interfaces on NFV and 5G solutions and enabling the interoperability across facilities. This proposed methodology aims to define how 5GASP SME solutions will be defined, modeled, designed, and implemented to create a NetApp ready to be automated, integrated, and tested in the envisioned CI/CD process. This methodology will be defined according to the requirements set by WP2 and to the ETSI standard of OSM/SOL006 (YANG model), as well as ONAP/SOL001. The generic process definition will help to enhance NetApps with solutions like the model-transformation service designed in WP3 and proposing assessment processes and KPIs of interest on the targeted vertical. Guidelines will also be provided to NetApp developers to help the KPI of service creation time by suggesting optimizations and best practices.

The document describes the onboarding procedure from a high-level perspective, showing the workflow of this process from the design phase to the onboarding and testing of the NetApp. In the same way, the location of the methodology in the 5GASP architecture is also discussed, together with the internal and external view of the onboarding service component.

To provide a general vision of the steps that a developer or an experimenter should take to prepare and onboard their NetApp, this deliverable will introduce the planned NetApp design and development methodology that will be followed in the 5GASP platform. To this end, the lifecycle of this procedure is presented and integrated within the global 5GASP methodology. Each phase of the process is discussed, specifically highlighting the design and development phases, while the testing and validation ones are more lightly detailed, as they are out of the scope of this document as WP5 deliverables cover them.

Finally, the NetApp onboarding approach is presented, describing how the NetApps will be managed in the 5GASP platform and the categories in which they can be divided, which are the NetApps considered as VNFs, and the NetApps considered as CNFs. Also, it is discussed how the NetApps will be defined with a combination of descriptors that represent the VNF itself, the network slice in which it will be located once deployed, and the testing and validation requirements that the NetApp will have to pass to be certified by 5GASP and to be available in the NetAppStore for the community.

1.2. Approach and methodology

As part of the WP4, this deliverable contains the initial efforts to transform the 5GASP solutions to NetApps that are ready to be automated and onboarded in the test facility sites via the testing and validation processes. The solutions modelled and designed by the SMEs will be developed in such a way that will allow interoperability across facilities, as well as their operation in inter-domain scenarios. By following this methodology, the NetApps will benefit from the enhancements designed in WP3, like the automated NetApp model translation. Also, the adoption of VNFs by the SME's solutions will follow the requirements set in WP2, taking into account the 5GASP architecture design. This deliverable will be produced when the tasks

T4.2 and T4.3 start, providing an initial methodology as an input for T4.2 and T4.3 that NetApp's developers should follow.

1.3. Document structure

The document consists of 5 chapters, being the first one the introduction, which presents the objectives of this document and the approach and methodology; and the last one the conclusions, which sums up the content introduced along the deliverable.
Chapter 2 introduces the NetApp onboarding procedure overview, showing the ecosystem workflow and the architecture integration and description of the onboarding component.
In Chapter 3, the NetApp design and development methodology are presented, including the NetApp onboarding lifecycle, and detailing the design and development phases.
The NetApp onboarding approach is introduced in Chapter 4, where the NetApp management procedure is discussed. The NetApp categories are differentiated, and the three descriptors that will define a NetApp are disclosed.

2. NetApp onboarding overview

This section provides an overview of the NetApp onboarding procedure proposed by the 5GASP project, describing and indicating the location of the onboarding process in the 5GASP architecture and the proposed workflow.

2.1. Architecture

2.1.1. Integration in 5GASP architecture

In order to achieve its goals, 5GASP proposes an approach which defines several functionalities and services and their interactions, as depicted in Figure 2.1, and which becomes the starting point for designing the global 5GASP architecture, including several main architectural functions, high levels design, usable in 5G networks, described by layered components:

- Validated NetApps catalogue in the marketplace, repositories
- NetApps DevOps/experimentation framework (CICD pipeline)
- NetApps services deployment orchestrator
- NFV (VMs/CNs) orchestrator (OSM/ONAP), YANG or TOSCA modeling descriptors (VNFD, PNFD and NSD) based on YANG or TOSCA
- 5G-capable facilities for 5GASP framework

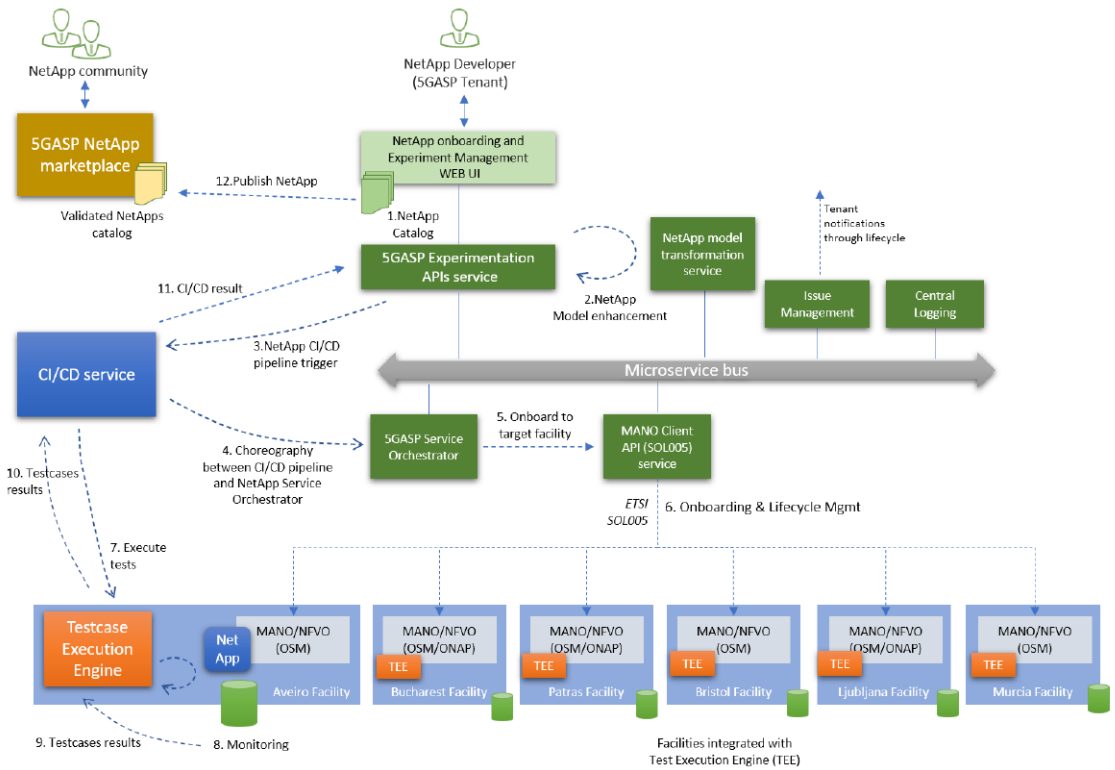

Figure 2.1. 5GASP approach on DevOps experimentation and certification readiness lifecycle

The NetApps onboarding in 5GASP requires introducing the DevOps concept in 5G networks and facilities, close to IT software release cycles, testing and validation.

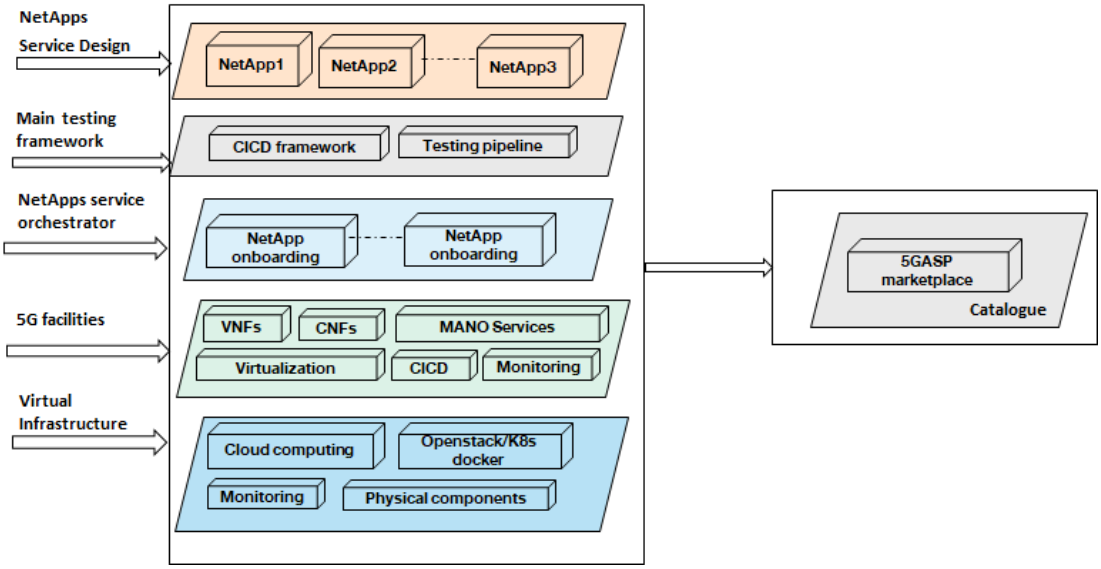

Figure 2.2. 5GASP architecture simplified overview

In a bottoms-up approach, as described in this simplified overview (see Figure 2.2), the 5GASP ecosystem architecture is based on a set of services and applications, VNFs or CNFs running on top of the virtualized infrastructure, apps hosted in VMs or containers, with support of different virtualization tools (e.g., OpenStack, K8s/Docker). The virtualized infrastructure is based on the previously 5G-PPP projects, as 5GEVE [1], 5GTANGO [2], 5GVINI [3]. From an architectural perspective, it enables each facility with cloud computing resources to be accessed by the required functions and NetApps through network softwarization. The 5G facilities support dynamic services instantiation and termination, cloud resources allocation, flexibility and services allocation in cloud infrastructure. Each facility uses a set of open monitoring tools as telemetry data from NFVI, Orchestrators or different 5G system components.

The 5G architecture should integrate DevOps functionality, supporting the NetApps onboarding requirements for VNF life cycle management:

- The initial setup instantiation (Day 0): configuring the VNF to provide the expected service
- Service initialization (Day 1): defining the required configuration for service initialization
    - Example: Python scripts (NETCONF/YANG); Ansible playbooks, etc.
- Runtime configuration (Day 2), applicable for both VNFs or CNFs: configuring the VNFs during runtime
    - Example: Python scripts; Ansible playbooks, REST APIs, etc.
- Closed-loop operation for auto-scaling and auto-healing

OSM onboarding for functions deployment (Day1/Day2) is described by charms scripts (code that performs a desired functionality, to be deployed just after instantiation time, or time after on demand), integrated into network functions descriptors/packages, compliant with

ETSI SOL006 standard, defining the VNF components, the NFVI requirements (RAM/CPU/storage) and compute performance attributes (CPU Pinning, NUMA, huge pages size, performance attributes as SR-IOV). With ONAP, through ETSI SOL001 TOSCA, the VNFs/CNFs design and onboarding process is composed of some steps for services instantiation and resource management through the ONAP end-to-end design services capabilities for model and artifacts. Moreover, the 5GASP service orchestrator links the NetApps onboarding process to targeted facilities through open APIs for ETSI NFV SOL 005 - RESTful protocols specification for the Os-Ma-nfvo (ONAP through SOL005 Adapter that supports NS/VNF Package Management and NS LCM) [4].

To achieve DevOps principles in 5G networks for network automation, fast service delivery and to offer improved collaboration between services owner and service providers owners, the CI/CD pipelines are introduced for tests execution and validation in each test case execution engine facility (see Figure 2.3).

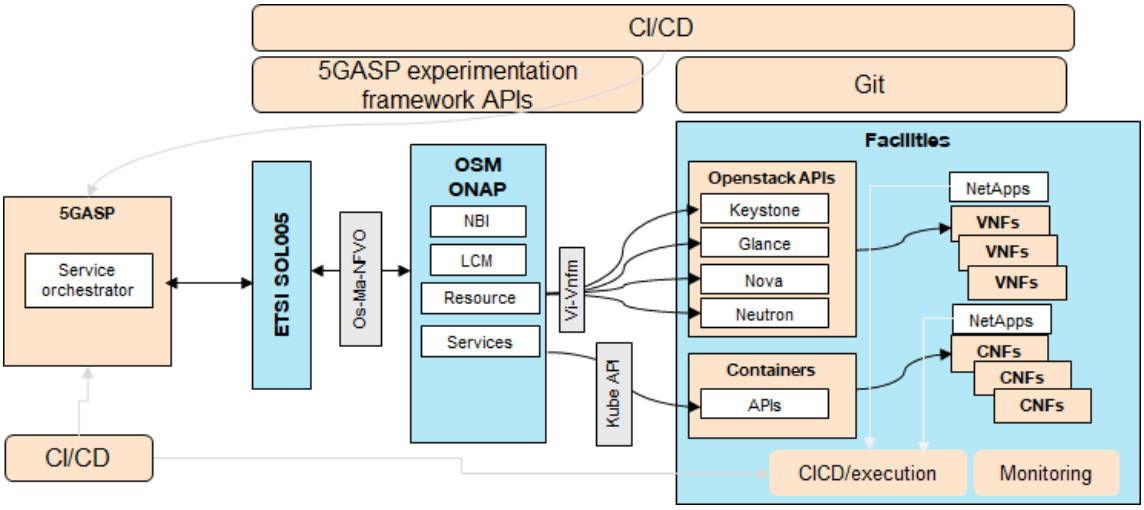

Figure 2.3. 5GASP DevOps architecture

Figure 2.4 shows the 5GAPS deployment procedure and CI/CD pipeline. The onboarding is performed before the pre-flight tests that are used to check the validity of the submitted descriptors. In fact, the onboarding itself is the event in charge of triggering the NetApp deployment and the CI/CD pipeline.

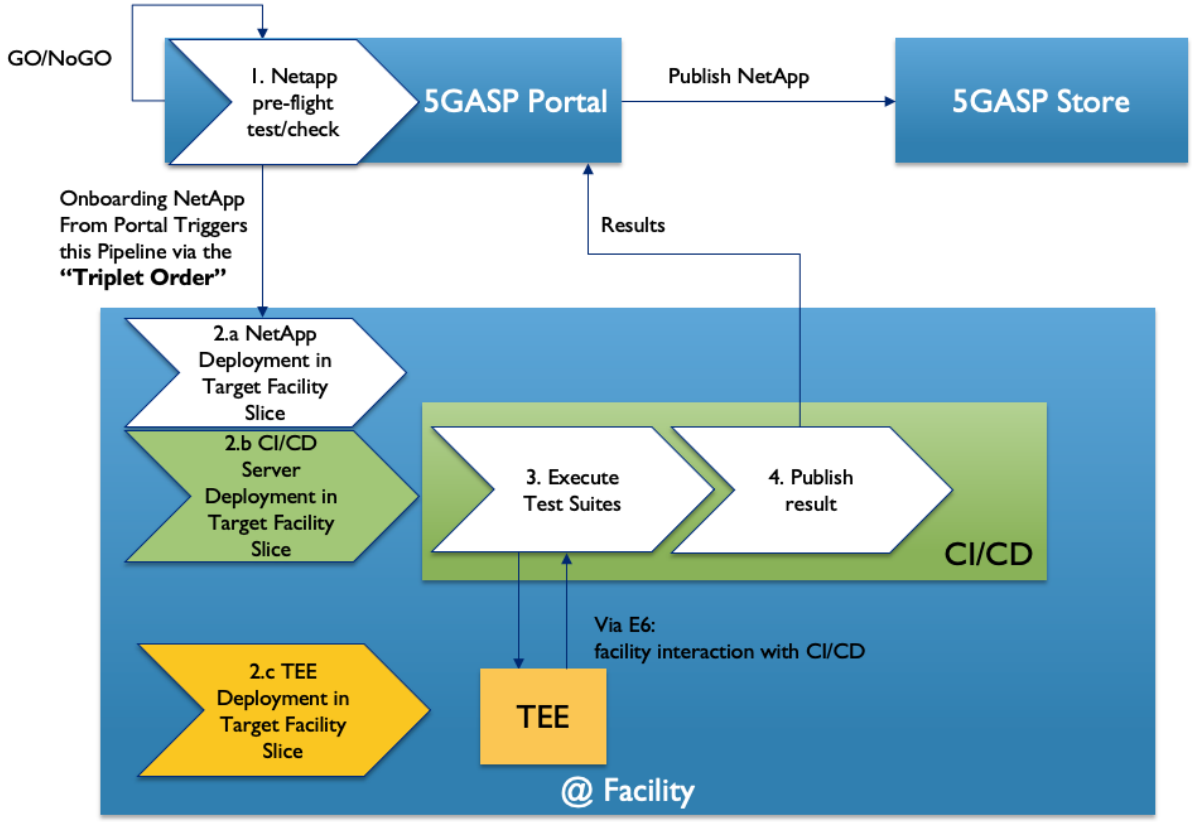

Figure 2.4. 5GASP deployment and CI/CD pipeline

2.1.2. Onboarding configuration

The 5GASP portal aims to provide an effortless way for the NetApp developer to onboard its NetApp. To that extend, the portal solution will be based on an open-source project, Openslice [5]. Openslice will present a single point of entry to the 5GASP system for the developer and a straightforward procedure towards the onboarding process. Furthermore, it offers user-friendly UI, multi-tenancy, and support for onboarding VNFs to target facility NFVO and an Open API based on TMFs APIs to facilitate the aforementioned procedure.

To support this aim, a developer would be provided with an entry point (sign-in portal), and following its registration, the onboarding process can then be initiated. Within the project's scope is the introduction of an abstraction layer in the form of a unified Experimental Model that would tone down the complexity of the onboarding procedure for an inexperienced portal user. The Unified Experimental Model is composed of a triplet bundled together as a single entity, composing the service deployment order. The segments comprising the unified entity are depicted in Figure 2.5 and are expressed by the TMF's ServiceSpecification [6] resource model. The introduction of ServiceSpecification class allows each part of the triplet to be seen as a service that can be instantiated based on a given set of characteristics and tracked throughout its lifecycle by the developer.

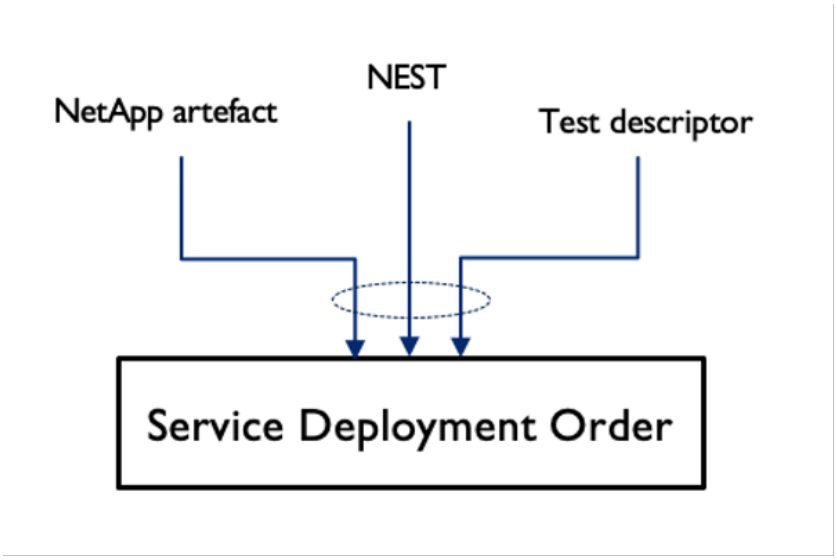

Figure 2.5. Unified Experimental Model

A non-exhaustive reference of the triplet's entities follows as they were extensively covered in D2.1 [5]. First, a NetApp artefact, described as VNFs/NSDs, is onboarded in the respective repository expressing the resource aspects of the NetApp. Next, the prerequisite network template, namely the NEST (see Section 4.1.2.2), is selected to ensure that the required network requirements for the NetApp's deployment are met. Finally, the testing process is outlined by the introduction of the testing descriptor entity that can be either selected by a pool of pre-defined testing pipelines or, alternatively, onboarded by the developer accommodating custom scenarios and iterations.

In the end, all the aforementioned segments are bundled under the scope of a common Service Deployment Order. In this regard, TMF's Service Order [7] model is employed. Following a conventional order capture and fulfillment pipeline, once an order is placed, the service fulfillment process is instantiated and runs the delivery process as per the requested operations onto an administrative domain. Meanwhile, the fulfillment process is tracked by the developer, who is notified throughout the entire procedure.

2.2. Ecosystem workflow

The onboarding procedure in the 5GASP system involves the uploading to the 5GASP portal of the NetApp, by means of the triplet of descriptors that defines it. The NetApp developer performs the onboarding itself (or by a NetApp experimenter), and the interaction is done against the unique 5GASP portal.

The process will be totally interactive with the user, requiring more actions in the first stages and evolving towards a more automatic procedure further on in the project. This is because the automated functions will be integrated gradually.

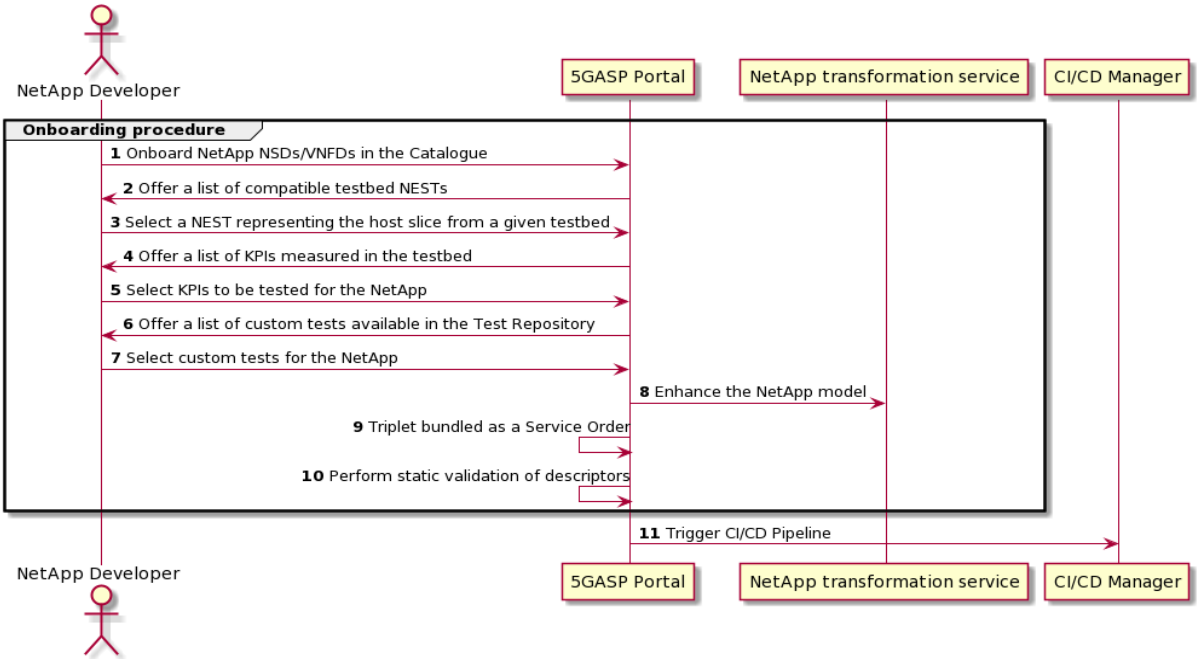

Figure 2.6. Onboarding and connection to CI/CD service

The onboarding procedure is depicted in Figure 2.6. This figure shows the different steps that conform the onboarding in the initial stages of the 5GASP project, in which the NetApp descriptors will be uploaded one by one. By doing so, the developer will be able to interact with the 5GASP project to configure different NetApp and testing parameters in each step.

It all starts with the upload of the NetApp NSDs/VNFDs, which will be stored in a catalogue. Then, based on the submitted NSDs and their respective network requirements, the developer should select the test site that fulfills the latter. In advanced project phases, the portal could compare the network requirements with the capabilities offered by the multiple facility test sites, filtering out the unfitting ones or even automatically selecting the ones that match the criteria of the NetApp. The list of test sites is shown to the developer as a list of NESTs representing each facility, and each one defines the host slice that will be reserved if the site is finally selected. The developer chooses one of them, and this information is stored together with the NetApp NSDs/VNFDs. Alternatively, the developer can select among a list of pre-defined NESTs, offered by the 5GASP portal, the one best suited to the NetApp. Based on this selection, the 5GASP portal offers the developer a list of KPIs that can be measured in the NetApps deployed in the corresponding facility test site. Then, the portal offers a list of custom tests available in the Test Repository, which the developer may select to test the NetApp. These custom tests can be pre-defined and available in each facility. In later project phases, the developer of the NetApp, apart from the default testbed tests, should be able to upload its own test suites (in the form of custom test scripts or test VNFs) that could extend the available test cases to be run against.

At this point, the NetApp is sufficiently defined in the 5GASP portal, and it is composed of the three descriptors: (i) the NSDs/VNFDs, (ii) the NEST, and (iii) the tests descriptors. Now, the descriptors will be sent to the NetApp transformation service, which will automatically enhance the descriptors, looking for bad practices and trying to correct them. Once this is

done, the triplet of descriptors can be bundled as a TMF Service Order [7] and pass a static validation of the descriptors. If succeeded, the onboarding procedure is complete.

Finally, the 5GASP portal can trigger the CI/CD pipeline interacting with the CI/CD Service Manager and send the required information for the deployment and the testing. The details about these subsequent testing phases of the 5GASP workflow are covered by WP5 and will be initially examined in D5.1.

3. NetApp design and development methodology

This section details the NetApp design and development methodology that will be followed by the 5GASP project. The objective of defining a design and development methodology is motivated by the need to bring the NetApp developers and the 5G ecosystem together. In this way, the methodology will reduce the time and cost for making new 5G NetApps or adapt existing ones to these new technologies.

3.1. Lifecycle

5GASP Lifecycle is a process used by the NetApp developers to design, develop and test high quality NetApps. This lifecycle aims to produce a high-quality NetApp that meets or exceeds customer (for example, MNO) expectations, reaches completion within times and cost estimates. 5GASP lifecycle includes four main phases: design, development, testing, and validation (see Figure 3.1).

Figure 3.1. 5GASP lifecycle

A high-level description of these phases is presented below. They will be defined and described in the appropriate WPs and deliverables in more detail. Design and development are described in this document, whereas the testing and validation will be defined in WP5.

3.1.1. Design

The design of a NetApp consists of three main steps, as they are depicted in Figure 3.2.These steps are necessary to be completed by the programmer so that their existing code can be onboarded as a NetApp through the 5GASP platform and tested through 5GASP's testing framework.

Figure 3.2. Phases of the design of a NetApp in the context of 5GASP

The design phases shall be elaborated in Section 3.2 of the document.

3.1.2. Development

Following the design phase, the developer can start developing the NetApp. The development consists of materializing the design considerations that have been studied in the previous step, resulting at the end in a NetApp ready to be onboarded to the 5GASP portal. Consequently, the development of the NetApp is also divided into three steps (see Figure 3.3), one for each component of the needed triplet: the VNFDs/NSDs, the network slice and the test descriptors.

Figure 3.3. Phases of the development of a NetApp in the context of 5GASP

The development phase is further discussed in Section 3.3.

3.1.3. Testing

Testing is a very important and required phase of the lifecycle, which aims to check whether the developed NetApp matches expected requirements and to ensure that NetApp is defect-free. It involves the execution of NetApp components using manual or automated tools to evaluate one or more properties of interest. The purpose of NetApp testing is to identify errors, gaps, or missing requirements in contrast to actual requirements, which are pre-defined for each NetApp on the first stage of the overall 5GASP methodology. In 5GASP, the testing phase will include the next stages (see Figure 3.4).

Figure 3.4. Testing stages of a NetApp in the context of 5GASP

NetApp and test VNFs deployment in test environment stage includes:

- Scheduling of the tests.
- Deployment of NetApps and test VNFs.

During the Test Execution stage, the system will run the included, selected and/or necessary tests to ensure the well-functioning of the NetApp. These tests will be created by the NetApp owners or will be selected by them from an available repository in the platform, to test the proper functioning of the NetApp.

After this, the following procedures will be performed during the Monitoring and analysis stage:

- Monitor infrastructure, NetApps and test VNFs
- Analysis of gathered data
- Evaluation of metrics
- Extraction of KPIs

Outcomes of this stage will be used for the Test validation stage (see Section 3.1.4). All the testing procedures will be defined and discussed in detail in WP5 deliverables.

3.1.4. Validation

One of the biggest challenges in upcoming 5G DevOps environments is the validation and verification (V&V) of virtual network functions (VNFs) and network services (NSs) against different execution platforms so that service operators can be sure that VNFs and NSs behave as expected immediately after they are deployed and put into production [8].

After Testing phase Validation process takes place. Validation can be defined as the process of evaluating the developed NetApp during or at the end of the development process to determine whether it satisfies specified requirements (defined KPIs) [9].

Several KPI could be extracted from 3GPP/ETSI/others covering:

- Functional Suitability (Functional Completeness, Functional Correctness, Functional Appropriateness);
- Performance efficiency (Time-behavior, Resource utilization, Capacity);
- Compatibility (Co-existence, interoperability);
- Reliability (Maturity, availability, Fault tolerance, recoverability);
- Security (Confidentiality, integrity, non-repudiation, accountability, authenticity);
- Maintainability (reusability, analyzability, Modifiability, testability);
- Portability (adaptability, instability, replaceability).

As the issue of validation is critical, ETSI NFV ISG has released a number of documents approaching the need and the framework for pre-deployment validation and testing, as well as interoperability and portability [10].All these and other documents will be analyzed in WP5, and the needed procedures will be defined for the above mentioned KPIs.

3.2. Design

For a NetApp developer, in order for their codebase to be onboarded and tested in the 5G environment of 5GASP, several 5G-specific designs must be undertaken by the programmer, so that the 5GASP environment is able to 'understand' and then deploy and test the software under consideration.

The first design consideration towards producing a NetApp based on a custom code is to define the main components of the NetApp in terms of services (described with Network Service Descriptors-NSDs) and interworking service components, that is Virtual Network Functions (VNFs) in the 5G terminology. This practice should be familiar to engineers of cloud-bound applications, as they do use the notion of services for designing and building their applications. Therefore, the programmer must list the services that are part of their overall application and then describe these services with NSDs. This means that each service must be described by an NSD as there exists a 1-1 relationship between a service and an NSD.

Overall, in this first NetApp design step, the programmer must clearly define the list of NSDs and VNFs as well as their respective cardinality since the latter is important for a testbed in order to be able to accommodate them. As an example, the programmer shall communicate to the 5GASP platform that their application consists of *N* NSDs and *M* VNFs where an NSD can consist of more than one VNF, since there is a 1-many relationships between an NSD and a VNF as is the case of programmer's services: one service is composed by one or more service components/functions.

During this step of the design of the NetApp, the goal is that the programmer actually defines the NSDs for all services of the NetApp. The NSDs shall include at least the following important information that shall be taken into account by 5GASP's control plane when the NetApp will be onboarded:

- Number of VNFs that are part of the service.
- Type of packaging, i.e., if the service shall be packed in a VM-based manner or in a Container-based manner (VNF/CNF, as we shall elaborate later in this document).
- Maximum required latency value in milliseconds.
- The need for Internet connectivity.
- Hardware resources required to deliver the service: in terms of CPU cores, RAM and hard disk requirements.
- The service delivery model: i.e., if it is packaged as a VM image, a docker container in DockerHub, etc.
- Ingress and egress bandwidth requirements of the service.

This first design phase of the NetApp also includes the definition of dependencies from other NetApps. This is especially important in the implementation of the 5GASP use cases, which consist of more than one inter-working NetApps, that work together to meet the technical objectives of the use case.

This second design phase encompasses the definition of the 5G network requirements of the programmer's services. This is implemented with the programmer defining how many network slices are needed for their needs. As briefed earlier in this document, the slice(s) definition shall adhere to the NEST template, which is explained in detail in Section 4.1.2.2 of the document.

The third design phase of the NetApp, before it can be submitted to 5GASP for onboarding, is to define the sets of tests that must be undertaken by the 5GASP platform in order to ensure that the NetApp does not exhibit any errors during instantiation and/or operation over the relevant 5GASP testbed(s) where the NetApp shall be onboarded. Tests shall be defined according to Test Descriptors as elaborated in Section 4.1.2.3.

3.3. Development

Once the design of the NetApp is performed, the developer can start developing the different descriptors that compose the NetApp as a whole. To do this, the methodology considers a division of the development phase into three steps: (i) the VNFDs/NSDs, (ii) the GST/NEST, and (iii) the test scripts and test VNFs descriptors. The development insights of the NetApp functionality itself, i.e., the application code, are out of the scope of this methodology definition.

The first descriptor to take into account is the VNFDs/NSDs that compose the NetApp. These descriptors must follow the standards that will be supported by the NFVOs offered by each one of the test sites (OSM8,9, TOSCA). This information will be available to the experimenters to avoid problems with the descriptor structure and keywords.

Depending on the type of NFVO to be used, the descriptors' format will vary, but the developer has to take into account the version of the NFVO, as there may be differences between them. For example, the descriptor format for OSM Release NINE is not the same for OSM Release EIGHT (or previous releases), as they implement a new northbound API to be upgraded to the new ETSI SOL006. To prevent the developer from having to be aware of these details, 5GASP will offer the specific type and version of the NFVO available on testbeds, as well as examples for them. Furthermore, some examples of descriptors will be available to simplify the developer's job, easing the task avoiding their creation from scratch. It could also be possible to offer some "ready-to-fulfil" descriptors, which could be customized by developers (for example, with the number of hardware resources or network interfaces). Also, a list of network and computing resources available on each facility will be offered to developers, where they can easily select the resources they want to use in their NetApps.

When preparing the GST/NEST descriptor for the NetApp, it is important to take into account the capabilities of the testbeds. In the first stages of the project, a list of pre-defined NEST for each testbed will be offered to the developers when onboarding the NetApp. Therefore, the network slice assigned to the NetApp in the deployment will be one of the default network slices. Later on during the project, the experimenter will select the desired capabilities for the network slice. Then, a series of available NESTs from the different testbeds will be offered, leaving the decision of which one will be used to the developer. Finally, in the final stages of the project, the portal will also accept the NESTs defined by the experimenter, choosing a

best-effort option if the requirements demanded by the NetApp cannot be fulfilled by any testbed.

As a consequence, it is important for the developer to know the network resources that the NetApp will demand once deployed to achieve an optimal operation and pass all the test and validation procedures and obtain the certification by the 5GASP platform. Therefore, in the following, we enumerate some example aspects to consider when fulfilling the NEST template:

- Area of Service and Region Specification: the preferred location to deploy the NetApp; this has to be considered if the test facility site of the chosen area has the necessary resources to host the NetApp properly, e.g., if the test site has edge capabilities.
- Throughput: necessary to estimate the bandwidth use that will require the uplink and downlink channels of the NetApp.
- Isolation level: related to the slicing resource isolation, it is important to consider if the NetApp can share physical resources with other applications.
- Mission-critical support: whether the network slice host critical services.
- Slice QoS parameters: indicates the kind of traffic that it will be hosted by means of the 3GPP 5QI standard.
- Maximum PLR: the maximum Packet Loss Ratio that the NetApp could handle without disrupting the service.
- Supported device mobility: it is important to consider the effects of speed on wireless communications in the automotive vertical, as the radio link technology and capabilities will have to support it.

Finally, regarding the development of the testing descriptors, depending on the type of tests, the approach may change. Initially, multiple infrastructure-related tests will be pre-provided by the different test facility sites. Thus, the developer can avoid implementing these kinds of tests, as they will be available to select during the onboarding process. They will have the correspondent KPIs associated to be included in the NetApp descriptors.

Moving on to the custom test scripts, the code itself is under the responsibility of the developer, and it will need to define an output to establish the success or failure of the test. Moreover, it is still to be discussed and decided in the project the details about the test repositories where these kinds of tests will be located.

Regarding the custom test VNFs, the development considerations are similar to the ones presented and discussed above in relation to the NetApp VNFDs/NSDs. It is important to mention that from the point of view of these test VNFs, the NetApp must be considered as a BlackBox with some inputs and some outputs, which will be the ones used to validate the test. In this way, these VNFs will commonly embed specific applications with a certain configuration prepared to validate a concrete aspect of the deployed NetApp. For example, a testing VNF could be a traffic generator together with a certain data packet trace in order to evaluate the behaviour and response of the NetApp when that specific traffic is received and processed.

4. Onboarding approach

This section provides some more insights about the onboarding, concretely, regarding the NetApp management, the types of NetApps, and the descriptors that will define them. We also provide a technical example of the workflow from design and development to the onboarding of one of the NetApps presented in the project, the virtual On-Board Unit (vOBU).

4.1. NetApp Management

The project goal is to express the NetApps as a set of virtual functions to facilitate an automated, repeatable deployment and testing cycle. To that extent, NetApps should be transformed into widely utilized models that derive from major standards bodies. Therefore, the modelling and packaging of the ETSI's SOL005/OSM (YANG model) and/or SOL001/ONAP (TOSCA model) will be followed. Consequently, adaptions and enhancements could be made on those models to enable particular deployment and testing scenarios. Furthermore, given the *VNFication* phase of the NetApps, implementation can support both ordinary VM-based approaches and the more contemporary container-based ones. In the latest years, the global tendency aims towards the latter approach and 5GASP envisages not only to go along, but actively contribute to the development of the NetApp community. Thus, developers would be expected and strongly encouraged to conform to that approach.

Furthermore, 5GASP envisions the option to provide developers with a single entry-point by means of a portal to enable a straightforward procedure towards the onboarding process. This portal will allow any developer to onboard its NetApp, specify the accommodating testbed to host it, and describe the tests that should be triggered once the NetApp is deployed. This "triplet" is bundled together in a single entity, creating a unified abstraction for all sites, thus assuring that the onboarding, activation and testing can be properly performed on any 5GASP facility. Also, aiming towards interoperability with any NFV/3GPP compliant 5G System, the project's approach for each entity of the "triplet" is to be defined under widely embraced models in the industry, i.e., GSMA's GST/NEST, TMF's ServiceSpecification and ServiceOrder, as further presented in Section 4.1.2.

4.1.1. NetApps categories

Although NFV is designed to be completely agnostic about how VNFs are instantiated, in practice, the most common method is deploying them as virtual machines (VMs). Nevertheless, in recent years the instantiation of VNFs as Linux (docker) containers has become more popular, since its deployment, scaling, etc., is considerably faster and requires lower resources. However, for certain purposes, it may be necessary to use one or the other depending on the characteristics and needs of the NetApp (or even depending on the facility where it is going to be deployed), so the idea is to offer both possibilities transparently.

4.1.1.1. NetApps as regular VNFs

Deploying VNFs as VMs is the most widely used and common way to instantiate them. A VM-based VNF is a virtual machine with hardware resources, network interfaces, and essentially, an image (usually qemu-based) that includes/implements the desired functionality.

Therefore, one of the most important points when deploying a VM-based VNF is to have the image that conforms the VM, since it is the one that will contain the functionality that we want to offer in that VNF.

To do this, normally a previously configured image is available with basic functionality (for example, the necessary packages installed), ready to receive the specific configuration required (IPs, targets, configuration files, etc.) after instantiation. Thus, we have a generic image to adapt to the characteristics of the scenario, but specific enough for not having to do all the required configuration after instantiation (which reduces the time needed until the VNF is ready).

To automate the image generation, 5GTANGO [11] presents a method to create VM-based images, which consists of selecting a pre-existing container (or uploading one) that implements the required functionality. Once identified, it generates a VM with a vanilla OS (for e.g., a freshly installed Ubuntu distribution) and installs the container on it. Then, a new VM image is generated based on that VM, so an image with the required functionality is ready to be instantiated.

Another typical option is to use a base image and configure it when deployed using different methods (for example, Ansible). In this case, a yaml file specifies the packages and functions to be installed, defining with a template and scripting system for the configuration files to be deployed on the VNF (a VM in this case). Another possibility to inject configuration is using Day-0 and Day-1 configuration (from OSM) that uses Juju Charms to configure instances. As abovementioned, Day-0 is the configuration that applies while instantiation (cloud-init files, RSA public key, etc.), and Day-1 the configuration to be injected and applied once the VM is deployed (install apache, clone a repository, etc.).

4.1.1.2. NetApps as CNFs

Containerized Network Functions (CNFs) is another approach in building cloud-native 5G NetApps with expected low-latency and ultra-reliability guarantees, amongst other guarantees as deriving from the business specifications for NetApps. The goal in using the CNF approach is to move from the 'heavy' VM-based approach discussed in the previous section to the cloud-native CNF approach, where CNFs are composed of microservices, often as open-source and which are hosted in Containers, such as Docker containers.

The motivation behind the CNF approach is that monolithic Network Functions implemented in VMs take a lot of time to deploy and to upgrade on one hand, and on the other hand, their performance might be as much as the performance of the lightweight CNF approach in an operational 5G setting given the ability of CNFs to scale easily and fast, leveraging Container as a Service tools such as Kubernetes (K8s) which have many capabilities in detecting failures of microservices and performance degradations and automatically deploys a new microservice in a container. This allows the capability of having a new microservice that replaces the old one which has the required capacity.

Overall, the small footprint of microservices (note that a container may include one or more microservices) and their fast start times provides a high performance and scalable approach

to deal with the 5G requirements in terms of latency, reliability, etc. in many use-cases. However, engineering of microservices/CNFs for 5G use cases is not a tedious task. For this reason, designed principles such as Istio [https://www.istio.io] have emerged to help developers solving the problems that arise while building CNF-based NetApps. The cloud-native industry is quickly evolving, and it is important that developers choose the proper tools and update their tools to derive CNF-based NetApps that meet their requirements in a real 5G deployment. A good overview of state-of-the-art tools for building cloud-native NetApps can be found at https://landscape.cncf.io/.

In terms of standardization of the CNF approach for building 5G networks and services, the 3GPP Technical Specification Group on Service and System aspects, has proposed an architecture in 3GPP TS 23.501 [12], which is based on services. Specifically, the architecture defines CNFs and reference points that are used to interconnect CNFs. As shown in Figure 4.1, with this service-based approach, the 5G network employs a service-based architecture which uses services that communicate with each other through a message bus where services are registered and subscribe to each other.

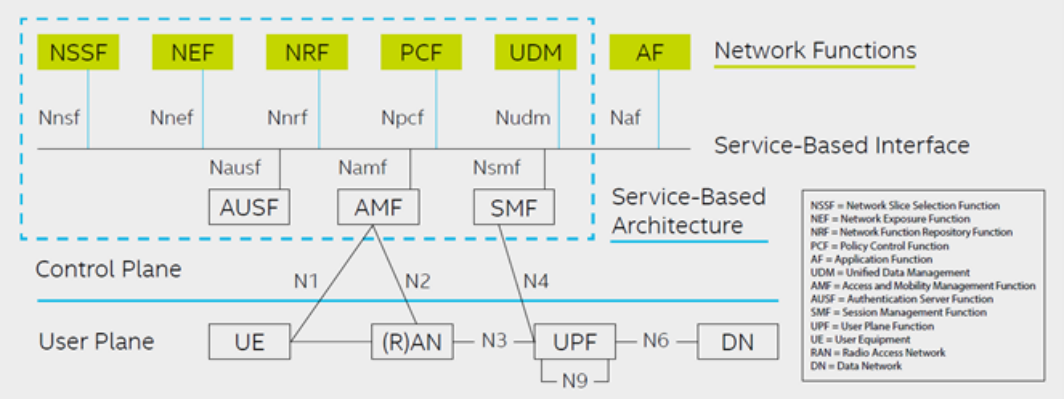

Figure 4.1. Proposed 3GPP architecture for cloud ready Network Functions

The goal of this stream of work of 3GPP is that 5G networks can ultimately be cloud ready and leverage the advantages of cloud-native approaches that are state-of-the-art in cloud service solutions. The business goal is to lessen the risk that 5G service providers lose customers if the latter chose to work with other players such as established cloud service providers. Therefore, 5GASP embraces the CNF approach and will demonstrate how selected 5GASP NetApps – such as the PrivacyAnalyzer NetApp by Lamda Networks - can be written as CNFs and be orchestrated via Kubernetes in selected 5GASP testbeds, such as the testbed offered by partners ITAV, UoP, Odins and UnivBris.

4.1.2. NetApps descriptor

The idea for 5GASP is to create a "meta-package" that includes the Network Slice requirements (NEST information), the NSDs that conforms the NetApp (with their respective VNFDs), and the tests the user wants to perform over them to validate the NetApp and obtain the certification. These packages can be completely included in the meta-package, or they can be references to already upload packages. However, in the first stages of the project,

these descriptors will be added individually and uploaded one by one, while the platform acquires the maturity to implement the meta-package management.

4.1.2.1. NSD/VNFD Packages

Besides the descriptor (the YAML/TOSCA file, which includes the description of the NS/VNF respectively), the packages can include other relevant information, for example, the image to be used, cloud-init files, some after-deployment configuration, such as charms in OSM packages or scripts following the TOSCA schema, etc. These files must be included in the package, following the established format for the orchestrator they are built for (as it is not the same package or descriptor for OSM as for ONAP).

As previously said, the idea is to work with a meta-package that is compatible with both types of descriptors (OSM and TOSCA, as 5GTANGO does). The meta-package would indicate which kind of descriptor each one is (for example, OSM and which release), and afterwards, based on this meta-package, the descriptors for the specific orchestrator would be generated. Nevertheless, in the first stages of the project, both packages will be formed independently and separately, depending on the testbed — or orchestrator — to be used.

Depending on the type of orchestrator required (OSM or ONAP), the format of the descriptor will vary, likewise in accordance with the used version of the orchestrator (it is not the same format between OSM Rel EIGHT than OSM Rel NINE, as the first uses ETSI SOL005 and the second ETSI SOL006). In this way, proper verifications must be performed in order to ensure the correctness of the package. Package examples are available in the orchestrator's repositories, as well as they will be in the 5GASP repository to be used as a template. Thus, developers won't need to create their packages from scratch, as they will have a guideline to perform it.

4.1.2.2. GST/NEST

Network slicing is one of the key enablers of the development of 5G technologies. 3GPP defined it as a dedicated logical network that provides specific network capabilities and network characteristics. These slices are usually bonded to a Service Level Agreement (SLA) agreed between a Network Slice Customer (NSC) and a Network Slice Provider (NSP). A network slice can be instantiated across multiple parts of the network, and it is composed by a set of dedicated or shared resources. If dedicated resources are used, the slice can be isolated from other network slices.

The Generic Network Slice Template (GST) is a set of attributes that can characterize a type of network slice or service, it is generic and is not tied to any specific network deployment [13]. This template was introduced by the GSM Alliance (GSMA) with the aim of introducing guidelines for verticals on how to address their service requirements and to facilitate the establishment of SLAs between operators and business customers.

The NEtwork Slice Type (NEST) is a GST filled with values. This set of attributes with certain values comply with a given collection of requirements derived from a use case defined by a network slice customer. The NEST is used as an input for the preparation of a network slice

instantiation performed by the network slice provider. Furthermore, multiple network slice instances can be created out of the same NEST, and existing instances can also be reused.

In Figure 4.2, the network slice lifecycle is illustrated. The NSC provides the requirements of its particular use case to the NSP. Then, the latter generates a NEST by mapping these service requirements into the attributes of the GST.

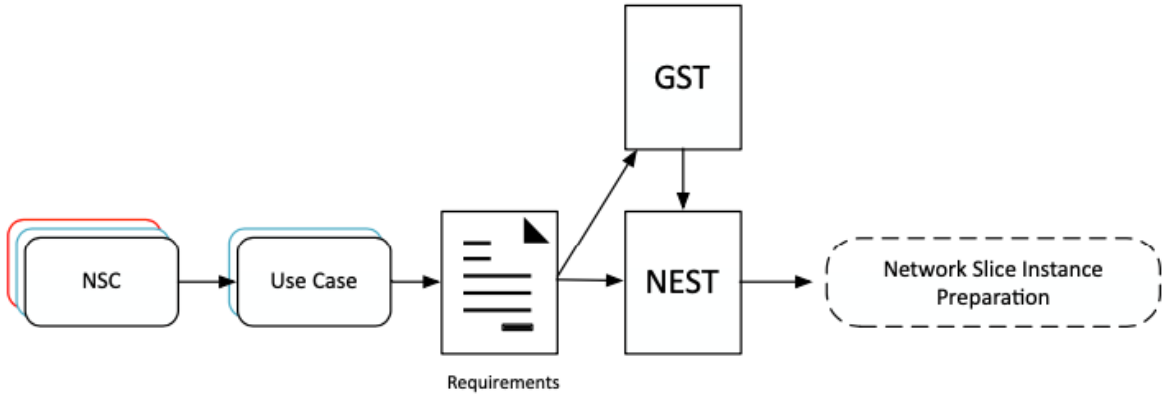

Figure 4.2. GST Network slice lifecycle

GST/NEST is the selected template to be used as the network slice descriptor in the NetApps definition. Each designated testing facility of the 5GASP project will provide information about the network requirements they support by means of NESTs. Therefore, the template will be perfectly aligned with this GSMA standard. By doing so, two different approaches can be followed by NetApps developers. The first one is to define and fulfill the NEST associated with its NetApp and including the network requirements that they think the NetApp will demand; and the second one is to select a pre-defined NEST which is associated with a certain 5GASP facility. While the latter is the best starting point and do not raise any issues, the former has to be handled with care, as the facility test sites need to support and implement the network requirements defined by the NESTs. Thus, in the case that none of the sites can fulfill the required network resources as a whole, a best-effort policy will be followed, and the testbed with the most similar network characteristics will be selected. By doing so, the design and development phase of the NetApp is simplified from the point of view of the experimenter.

In this way, in an earlier stage of the project, whenever a developer wants to onboard a NetApp, it may select the site that will host the NetApp deployment from a pre-defined list that will show the network specifications and characteristics of each of the facilities. Thus, the network slice descriptor of the NetApp will be a NEST with the selected site capabilities. Further on, the 5GASP system will automatically select the most appropriate site (or sites) to deploy the onboarded NetApp, based on the network requirements introduced by the developer. If there is more than one option, the developer will be able to select the one with a better match.

An example of a NEST from a NetApp of the Project is shown in Table 4.1. It corresponds to the NetApp 4: Multi-domain Migration NetApp.

Multi-domain NETAPP's NEST	
Area of service	SP, PT, UK, RO (AUTO-V use case)
Area of service: Region Specification	Murcia, Aveiro, Bristol, Bucharest
Downlink maximum throughput per UE	100 Mbps
Uplink maximum throughput per UE	100 Mbps
Isolation level	3: Tenant/service isolation
Slice quality of service parameters: 3GPP QoS Identifier (5QI)	9
Maximum Packet Loss Rate	1%
Supported device velocity	3: Vehicular: 10 km/h to 120 km/h

Table 4.1. NetApp4: Multi-domain Migration NetApp's NEST

4.1.2.3. Test Descriptor

To simplify the Test phase, 5GASP aims to design its own test descriptor, using as an example previous Test Descriptor files such as the 5GEVE [1] or 5GTANGO [2] ones. The concrete format of the descriptor will be specified in further phases of the project, though in this deliverable, the basis of what the descriptor must contain and its approached structure will be exposed.

In 5GASP, there will be two different types of tests: the tests included as scripts (basically code that will be executed) and Test VNFs. A test VNF is a VNF which purpose is to perform some test and that it's not involved in the functionality of the NetApp itself. In this sense, the difference between them is that the Test VNF is a separate VNF that is requested to execute the tests (as they are included inside the VNF), whereas the scripts can be executed inside the NetApp VNFs (the ones we want to test) or in a different place. It is also remarkable the possibility of using simple tests (available in the platform), such as iperf, ping, or similarly to check the basic connectivity and operation.

Moreover, the platform will include a test repository, with pre-defined tests and simple tests to be used, and the possibility of creating custom tests using the Test Descriptor. In this way, the main characteristic of the proposed Test Descriptor will be its ability to be divided into different steps or phases (5GTANGO alike), for example, "Setup", "Execution", and "Validation". The different steps are described below:

- Setup: This phase includes the definition of the scripts and the VNFs that will be used to perform the test, and also the deployment of the test VNFs. Regarding scripts, they could be included in the package as executable files, or the commands could be directly included inside the test descriptor, similarly as 5GEVE.

- Execution: This phase includes the launching of the scripts (both VNF and bare scripts, in the preferred order or simultaneously), and later the collection of metrics obtained from the executions.

- Validation: Finally, this phase includes the analysis of the obtained data, the computing of the KPIs using the obtained metrics as input, and lastly, the validation of both KPIs and tests.

However, these phases are still under study and consideration, and they may undergo some changes during the first stages of the project, as we may found adjustments that better fit the needs of the project.

4.2. NetApp Design, Development and Onboarding example

To illustrate the designed methodology presented in this deliverable, in this subsection, we provide an example of the application of this methodology to the design, development and onboarding of one of the NetApps that are proposed in the 5GASP project, the vOBU.

The vOBU NetApp proposes the virtualization of vehicle OBUs with the aim to create a MEC layer to offload heavy computational tasks from the vehicle and to serve data-access requests. By doing so, it provides the system with robustness against potential disconnections periods from the vehicle, it saves radio resources on the link and improves the data processing performance.

4.2.1. vOBU NetApp design

The first step to design the NetApp is to analyze its functioning and its architecture. This is because the programmer must clearly differentiate the multiple network functions and services that conform the application. In this way, in Figure 4.3, the architecture of the vOBU NetApp can be seen.

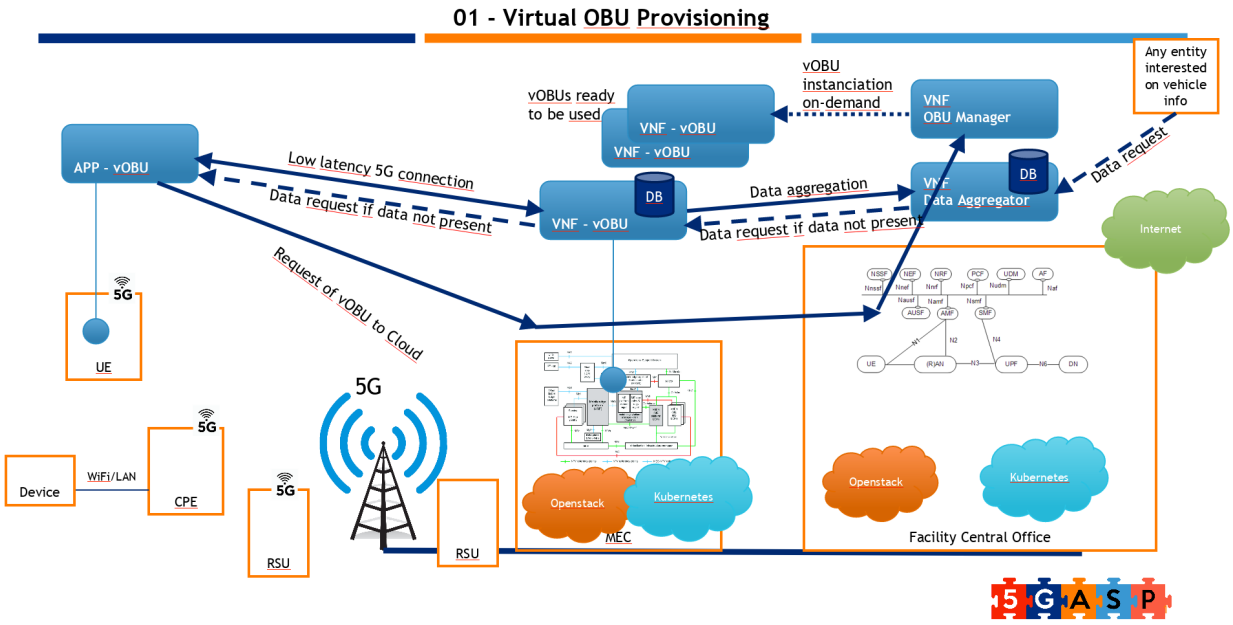

Figure 4.3. Virtual On-Board Unit (vOBU) NetApp architecture

Once the number of NSs and VNFs is decided, it has to be discussed the information the NSDs should include, taking into account the considerations detailed in Section 3.2. Some of the following items are examples of the kind of requirements that have to be defined in the design phase:

- Packaging: the VNFs will be deployed as VMs, ready to be used in OSM.
- Internet connectivity: yes.
- Hardware resources required: 2 CPU, 1 GB RAM, 10 GB HDD.
- Placement latency: 500ms.

The last step regarding the design of the NS and VNFs is to consider the dependencies with other NetApps. In this case, this NetApp do not have any dependency with other developments, however, in the vertical use case of 5GASP project, the vOBU NetApp will cooperate with other NetApps and that should be considered when designing the use case.

In the design of the network requirements of the NetApps, the developer has to define how many network slices are needed and the characteristics of them. In this case, the vOBU will simply need one network slice. Next, it has to be decided the requirements that will include the NEST template that will define the network slice on which the NetApp will be hosted. Among multiple parameters, below we present some of the requirements:

- Area of service and region specification: Murcia (SP).
- Isolation level: virtual resources.
- Mission critical support: inter-user priorization.
- Slice quality 5GPP 5QI: 69.
- Maximum PLR: 1%.
- Supported UE velocity: vehicular.

Finally, the last step of the design phase is the definition of the set of test that must be performed in the 5GASP platform against the vOBU NetApp in order to ensure a valid functioning and, if successful, resulting in the certification of the NetApp. As the testing procedure and its insights are currently under design and development, here we will focus on simple tests, mainly focused in the infrastructure, that will be used to validate the KPIs of the vOBU NetApp. To this end, a series of infrastructure tests will be defined to evaluate the metrics from which the KPIs are computed. These KPIs are the initial deployment time, the transaction speed of the messages, the PLR of the messages exchanged between the OBU and the vOBU, the service response time at the instantiation and the vOBU service downtime. These KPIs are shown in Table 4.2.

Generic KPI name	Metric Indicator (How)	KPI value	KPI unit
Initial time	Time needed to deploy for first time the entity (vOBU)	<5	Minutes
Transaction speed	Each message sent from an OBU needs to be redirected to the vOBU	500	Milliseconds
Packet Loss Ratio	Ratio of packets loss between the OBU and vOBU. Packets loss above packets sent	1	%
Service response time	Delay ratio while including vOBU instantiation	<30	%
Service downtime	Ratio of time the vOBU is not up and running	<10	%

Table 4.2. vOBU NetApp KPIs

4.2.2. vOBU NetApp development

Once the design phase has finished, the development of the NetApp can start. As the majority of the hard work has been already done in the previous steps, now it is time to reflect it in the multiple descriptors.

The first one is the NSD that defines the vOBU Network Service, it can be seen in Table 4.3. As previously mentioned, it includes the three VNFs that compose the network service and the network to which they are attached. This network is already present in the descriptor as it is known that the Murcia facility offer it. However, as previously discussed, in case the developer is unaware of the available networks of the test sites, a list of the networks ready to be used in each facility will be presented to the experimenters beforehand.

```
nsd:nsd-catalog:
  nsd:
  - constituent-vnfd:
    - member-vnf-index: '1'
      vnfd-id-ref: surrogates_mgmt_vnfd
    - member-vnf-index: '2'
      vnfd-id-ref: surrogates_vobu_vnfd
    - member-vnf-index: '3'
      vnfd-id-ref: surrogates_agg_vnfd
    description: Surrogates NS prepared for OSM version 8.
    id: surrogates_nsd
    name: surrogates_nsd
    short-name: surrogates_nsd
    version: '1.0'
    vld:
    - id: red800
      mgmt-network: true
      name: Red800BigNAT
      short-name: Red800BigNAT
      type: ELAN
```

```
    vim-network-name: Red800BigNAT
    vnfd-connection-point-ref:
    - member-vnf-index-ref: '1'
      vnfd-connection-point-ref: eth0
      vnfd-id-ref: surrogates_mgmt_vnfd
    - member-vnf-index-ref: '2'
      vnfd-connection-point-ref: eth0
      vnfd-id-ref: surrogates_vobu_vnfd
    - member-vnf-index-ref: '3'
      vnfd-connection-point-ref: eth0
      vnfd-id-ref: surrogates_agg_vnfd
```

Table 4.3. vOBU NetApp NSD

Next, we present in Table 4.4 one of the VNFs that compose the NetApp, the one corresponding to the vOBU entity itself. Here it is important to highlight some of the fields in relation to the design phase, such as the interface that will be connected to the network defined in the NSD, the image used to host the VNF and the resources required for that image. Here, the situation is similar to the network that is defined in the NSD, but with regard to the image and the flavors, in this VNF, the image and vm-flavor defined are known in advance, as they are present in the Murcia test site, although a list of offered images and flavors in each facility will be provided beforehand.

```
vnfd-catalog:
  vnfd:
  - connection-point:
    - name: eth0
      type: VPORT
    description: Surrogates vOBU entity prepared for OSM version 8.
    id: surrogates_vobu_vnfd
    mgmt-interface:
      cp: eth0
    name: surrogates_vobu_vnfd
    short-name: surrogates_vobu_vnfd
    vdu:
    - count: 1
      description: surrogates_vobu_vnfd-VM
      id: surrogates_vobu_vnfd-VM
      image: debian-baseSurrogates-certs
      interface:
      - external-connection-point-ref: eth0
        name: eth0
        type: EXTERNAL
        virtual-interface:
          bandwidth: '0'
          type: PARAVIRT
          vpci: 0000:00:0a.0
      name: surrogates_vobu_vnfd-VM
      vm-flavor:
```

```
    memory-mb: '1024'
    storage-gb: '10'
    vcpu-count: '2'
  vendor: SURROGATES
  version: '1.0'
```

Table 4.4. vOBU NetApp vOBU VNFD

Another thing to note in these descriptors is that they are designed and developed following the guidelines of OSM to be compatible with OSM Rel EIGHT, as the Murcia test site in which the NetApp is intended to be hosted counts with an instance of OSM8. In the same line, the supported OSM releases in each testbed will be exposed to the experimenters, as well as any other available NFVO.

Now that the NSD and the VNFs are defined, we can move on to the NEST. This is a straightforward step, as the heavy work has been performed in the design phase, analyzing the network requirements of the NetApp. Therefore, the only step here is to map them to the NEST template. In Table 4.5, the NEST template filled with the values discussed in the previous section is presented. The values are defined according to the guidelines of the GSM's GST (see Section 4.1.2.2). As in the previous case, a network slice able to fulfill these requirements (or almost identical) will be available in Murcia. In other case, the NEST would be selected from the list of predefined ones offered by the 5GASP portal.

vOBU NETAPP's NEST	
Area of service	SP
Area of service: Region Specification	Murcia
Slice quality of service parameters: 3GPP QoS Identifier (5QI)	9
Maximum Packet Loss Rate	1%
Supported device velocity	3: Vehicular: 10 km/h to 120 km/h

Table 4.5. vOBU NETAPP's NEST

Finally, the last stage of the development is the implementation of the tests. As presented before, 5GASP considers three types of tests: pre-provided infrastructure tests, custom test scripts and custom test VNFs. Due to the early stages of the project, at the moment there are only some infrastructure tests developed specifically for certain NetApps, thus they can not be considered as general pre-provided infrastructure tests.

As mentioned, for now there are only available some infrastructure tests prepared for the vOBU NetApp. These tests have been made ready to be used with Jenkins and Robot Framework. The latter is in charge of the test itself, which has been developed using Python, and the former's responsibility is to trigger the execution of the test. In this way, three different tests have been developed with the aim of evaluating three of the KPIs defined for the vOBU NetApp: the deploy time, the transaction speed and the PLR. In the following, we present as an example one of them, specifically, the one that tests the transaction speed between the OBU and the vOBU.

In first place, Table 4.6 shows the Robot test definition, which defines the test script in charge of obtaining the metric value and the condition to validate the test.

```
*** Settings ***
Library       TransactionSpeed.py

*** Test Cases ***
Testing the transaction speed between OBU and vOBU

    ${milliseconds}=    Transaction Speed
    Should Be Equal    ${milliseconds}    Less than 500 milliseconds
```

Table 4.6. Robot test definition

Secondly, Table 4.7 presents the Python script in charge of obtaining the average transaction time between the OBU and the vOBU. It is a simple idea that uses the ping command to obtain the values.

```
import paramiko, re

host = "1.1.1.1"
username = "jenkins-testing"
password = "password"

def transmission_speed():
    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    try:
        client.connect(hostname=host, username=username, password=password)
    except:
        print("[!] Cannot connect to the SSH Server")
        exit()

    stdin, stdout, stderr = client.exec_command("ping -c 5 1.1.1.2")

    pingResult = stdout.read().decode()
    regex = re.compile(r"(.*?) = (.*?)\/(.*?)\/(.*?)\/(.*?) ms")
    result = regex.search(pingResult)
    print(pingResult)

    if result:
        avgTimeTransmission = float(result.group(3))
        print("AVG:", avgTimeTransmission)
        if avgTimeTransmission < 500:
            return "Less than 500 milliseconds"
        else:
            return "More than 500 milliseconds"
    else:
        return "Not found"

if __name__ == "__main__":
    transmission_speed()
```

Table 4.7. Python test script

In the future, these tests will be generalized to be offered as pre-defined tests iin the facility test sites. Furthermore, more complex tests will be developed in the form of custom test scripts and custom test VNFs.

4.2.3. vOBU NetApp onboarding

With the development completed, it is time to onboard the NetApp. As explained in Section 2.1.2, the NetApp is composed by the three descriptors presented above. To enable the onboarding of them, 5GASP portal will offer an interactive procedure to upload the descriptors. Once done, the triplet will be converted into a Service Deployment Order and the CI/CD pipeline will be triggered. In this way, the procedure will be as follows:

1. Uploading of the NSDs and VNFDs to the portal.
2. Uploading of the NEST representing the required network slice. Although, at this early stage, the NEST will be selected from a predefined list.
3. Selection of the KPIs to be tested in the infrastructure for the NetApp.
4. Uploading of the tests. Although, at this early stage, the tests will be selected from a predefined list.

5. Conclusions

5GASP vision is to foster the use of NetApps in SMEs by introducing a well-defined approach to design and develop this kind of 5G applications. This proposed methodology will enable the automated and reproducible validation of NetApps across multiple facility test sites, including inter-domain scenarios. By doing so, the aim is to become an operational platform with strong industrial backup with the potential to be a reference ecosystem for the validation and deployment of 5G experiments.

In this project deliverable, the initial methodology for the design and development of NetApps has been proposed. The shape of this methodology has been done by using past test focused projects, such as 5GTANGO and 5GEVE, as foundation pillars. In this way, the ideas followed by the proposed methodology are an evolution of the different approaches that has been made in the past.

The initial methodology discussed in the document follows the requirements of the architecture that is being defined in WP2, and, at the same time, it will benefit from the enhancements that are being proposed and developed in the context of WP3. Moreover, the presented NetApp design and development preliminary workflows conform to the preparatory steps that connect with the testing, validation and certification phases that are proposed in WP5.

For that purpose, the placement of the onboarding methodology within the 5GASP general architecture and workflows is described and exemplified. Furthermore, an initial design and dissection of the onboarding steps to be taken to submit a NetApp to the 5GASP portal are also presented.

In addition, an initial description of the design and development of innovative NetApps is provided by emphasizing the information that a novel 5G developer should take into account when working with this type of applications.

Finally, the discussion is oriented towards the initial onboarding approach that will be followed by 5GASP experimenters. It is described how the NetApps will be managed and how they will be composed, using a triplet of descriptors to perfectly define each of the components. Also, an example of the application of the methodology is showcased using the vOBU NetApp.

The next steps for WP4 will focus on the iteration over this defined methodology, with the objective of improving it, evolving the procedures to adapt them to the 5GASP project necessities and receiving, at the same time, the feedback from the multiple SMEs participating in the project. At the same time, the WP4 future tasks will also work on the design and development of NetApps in the Automotive and PPDR verticals. Finally, the main focus of these iterations and enhancements to the methodology will be on the automation of the procedures, reducing the intervention and required knowledge from the 5G developers.

Bibliography

[1] "5GEVE," [Online]. Available: https://www.5g-eve.eu/.

[2] "5GTANGO," [Online]. Available: https://www.5gtango.eu/.

[3] "5GVINNI," [Online]. Available: https://www.5g-vinni.eu/.

[4] "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. V3.3.1," ETSI, 2020-09.

[5] 5GASP, "D2.1. Architecture, Model Entities Specification and Design," 5GASP, 2021.

[6] TMF, "TMF633 Service Catalog API User Guide v4.0.0," TMF, 2020.

[7] TMF, "TMF641 Service Ordering API User Guide v4.1.0," TMF, 2020.

[8] M. Z. e. al., "Verification and validation framework for 5G network services and apps," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2017.

[9] IEEE, "IEEE 610-1990 - IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries," IEEE.

[10] IEEE, "ETSI GS NFV-TST 001. Network Functions Virtualisation (NFV);Pre-deployment Testing; Report on Validation of NFV Environments and Services. V1.1.1," IEEE, 2016.

[11] 5GTANGO, "D4.1 First open-source release of the SDK toolset," 5GTANGO, 2017.

[12] ETSI, "5G;System Architecture for the 5G System (3GPP TS 23.501 version 15.2.0 Release 15)," ETSI, 2018.

[13] GSMA, "Generic Network Slice TemplateVersion 4.0," GSMA, 2020.