# 5Gtango

## D3.1 Verification and Validation strategy and Automated metadata management

<table>
<thead>
<tr>
<th>Category</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project Acronym</strong></td>
<td>5GTANGO</td>
</tr>
<tr>
<td><strong>Project Title</strong></td>
<td>5G Development and Validation Platform for Global Industry-Specific Network Services and Apps</td>
</tr>
<tr>
<td><strong>Project Number</strong></td>
<td>761493 (co-funded by the European Commission through Horizon 2020)</td>
</tr>
<tr>
<td><strong>Instrument</strong></td>
<td>Collaborative Innovation Action</td>
</tr>
<tr>
<td><strong>Start Date</strong></td>
<td>01/06/2017</td>
</tr>
<tr>
<td><strong>Duration</strong></td>
<td>30 months</td>
</tr>
<tr>
<td><strong>Deliverable</strong></td>
<td>D3.1 Verification and Validation strategy and Automated metadata management</td>
</tr>
<tr>
<td><strong>Workpackage</strong></td>
<td>WP3</td>
</tr>
<tr>
<td><strong>Due Date</strong></td>
<td>M4</td>
</tr>
<tr>
<td><strong>Submission Date</strong></td>
<td>2017/10/23</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>1.0</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>To be approved by EC</td>
</tr>
<tr>
<td><strong>Editor</strong></td>
<td>Mengxuan Zhao (EGM)</td>
</tr>
<tr>
<td><strong>Contributors</strong></td>
<td>Mengxuan Zhao (EGM), Franck Le-Gall (EGM), Philippe Cousin (EGM), Michael Christopher Woods (HWIL), George Xilouris (NCSR), Panos Trakadas (SYN), Anastasios Zafeiropoulos (UBI), Eleni Fotopoulou (UBI), Manuel Peuster (UPB), Stefan Schneider (UPB), Dimosthenis Kyriazis (UPRC), Maria Siapera (UPRC), Konstantinos Douloudis (UPRC), Evgenia Kapassa (UPRC), Patrick-Benjamin Bök (WEID), Marcel Müller (WEID)</td>
</tr>
<tr>
<td><strong>Reviewer(s)</strong></td>
<td>José Bonnet (ALT), Peter Twamley (HWIL)</td>
</tr>
</tbody>
</table>

**Keywords:**

V&V, DevOps, Catalogue, Monitoring, Metadata, SLA
**Deliverable Type**

<table>
<thead>
<tr>
<th>Deliverable Type</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Document</td>
<td></td>
</tr>
<tr>
<td>DEM Demonstrator, pilot, prototype</td>
<td></td>
</tr>
<tr>
<td>DEC Websites, patent filings, videos, etc.</td>
<td></td>
</tr>
<tr>
<td>OTHER</td>
<td></td>
</tr>
</tbody>
</table>

**Dissemination Level**

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU Public</td>
<td></td>
</tr>
<tr>
<td>CO Confidential, only for members of the consortium (including the Commission Services)</td>
<td></td>
</tr>
</tbody>
</table>

---

**Disclaimer:**

This document has been produced in the context of the 5GTANGO Project. The research leading to these results has received funding from the European Community’s 5G-PPP under grant agreement n° 761493. All information in this document is provided “as is” and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability. For the avoidance of all doubts, the European Commission has no liability in respect of this document, which is merely representing the authors’ view.
Executive Summary:

In 5G, to reduce the time-to-market for networked services and to lower the entry barrier to third party developers of Virtual Network Functions (VNFs) and Network Services (NSs) which are VNF graphs, an integrated Development and Operations (DevOps) methodology is crucial. One of the biggest challenges in DevOps is the Validation and Verification (V&V) of individual VNFs and NSs so that providers of these services can be sure of their behavior.

This document addresses the V&V strategy that 5GTANGO project will be following in the upcoming months. This V&V strategy is not only about the validation and verification testing process, but also about the complete chain that produces necessary input and manages or makes use of the output of V&V testing.

We begin this document by listing its main objectives, the main validation and verification targets and a high-level workflow of the V&V ecosystem. Then the architectural details of the V&V platform are provided, including the main stakeholders, the list of components and the possible configurations addressing different use cases. Two zooms on the architecture are presented: an external view, on how the V&V platform is framed by the 5GTANGO global architecture and an internal view, on how V&V functionalities are decomposed into modular functional blocks. The technical workflows are then presented followed by the methodology and approaches that 5GTANGO V&V platform will adopt. Details of the testing approach are also provided. As stated in the beginning, the DevOps approach also needs an efficient system of data collecting and management, it is indispensable to describe the monitoring and the policy and Service Level Agreement (SLA) management as data collection, as well as the VNF/NS catalogue with all the metadata as the metadata management.

This document tries to establish a high-level guideline for 5GTANGO V&V platform implementation, as well as a reference for the metadata management that is made available to the V&V platform.
Contents

List of Figures vii

1 Introduction 1

2 Overall Verification and Validation Strategy 2
  2.1 Ecosystem Workflow .......................................................... 2
    2.1.1 What does the Verification and Validation (V&V) Test and Why ........ 2
    2.1.2 The V&V and Supporting a V&V Ecosystem ............................. 5
  2.2 V&V Architecture ............................................................ 8
    2.2.1 Purpose of this Section ................................................ 8
    2.2.2 V&V in the Global 5GTANGO Architecture ............................ 9
    2.2.3 Workflow and Results Handling ....................................... 9
    2.2.4 The Three V&V Configurations ....................................... 9
    2.2.5 Architecture Description ............................................. 11
    2.2.6 Architecture and Component Details .................................. 13
    2.2.7 Supported Scenarios .................................................. 18
  2.3 V&V Methodology ............................................................ 19
    2.3.1 V&V Lifecycle ............................................................ 19
    2.3.2 Certification and Labelling Process .................................. 22
    2.3.3 Model-based Testing for Automatic Test Generation ................. 23
    2.3.4 Black Box vs White Box Testing ..................................... 25
    2.3.5 TTCN-3 Test Scripting Language ..................................... 26
    2.3.6 ETSI NFV Testing Methodology Alignment ............................ 27

3 V&V Testing Approach 30
  3.1 Tests Management ......................................................... 30
    3.1.1 Test Categories ....................................................... 30
    3.1.2 Considerations on Impact of Virtualisation on Testing Methods ...... 31
    3.1.3 Test Suite, Test Case and Test Profile .............................. 32
  3.2 Test Result Management .................................................. 33
    3.2.1 What Need to be Included in the Test Result? ......................... 34
    3.2.2 Presentation of the Test Result: a Star System ..................... 34
    3.2.3 Test Results Repository ............................................. 35

4 Monitoring in V&V 37
  4.1 Interplay of Monitoring Framework with V&V Platform Components ........ 37
  4.2 A Representative Example of Network Service Test Execution on V&V Platform 39

5 Test Execution Platforms 40
  5.1 Deployment Flavors ....................................................... 40
    5.1.1 Development Test Platform .......................................... 40
    5.1.2 Private Test Platform ............................................... 41
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.3</td>
<td>3rd Party Test Platform</td>
<td>41</td>
</tr>
<tr>
<td>5.2</td>
<td>Annotating Test Results with Platform Information</td>
<td>42</td>
</tr>
<tr>
<td>6</td>
<td>Catalogue and Metadata Management</td>
<td>43</td>
</tr>
<tr>
<td>6.1</td>
<td>5GTANGO Catalogue</td>
<td>43</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Goal of the 5GTANGO Catalogue</td>
<td>43</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Metadata</td>
<td>43</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Added Value Mechanisms</td>
<td>46</td>
</tr>
<tr>
<td>6.2</td>
<td>Policies Management Framework</td>
<td>48</td>
</tr>
<tr>
<td>6.3</td>
<td>SLA Management Framework</td>
<td>50</td>
</tr>
<tr>
<td>7</td>
<td>Conclusion</td>
<td>52</td>
</tr>
<tr>
<td>7.1</td>
<td>Scope and Architecture</td>
<td>52</td>
</tr>
<tr>
<td>7.2</td>
<td>Methodology and Approach</td>
<td>52</td>
</tr>
<tr>
<td>7.3</td>
<td>Catalogue and Metadata Management</td>
<td>52</td>
</tr>
<tr>
<td>7.4</td>
<td>Future Works</td>
<td>53</td>
</tr>
<tr>
<td>8</td>
<td>Acronyms</td>
<td>54</td>
</tr>
<tr>
<td>A</td>
<td>Bibliography</td>
<td>56</td>
</tr>
</tbody>
</table>
## List of Figures

2.1 Network Service Under Test and Metric collection points ..................................... 3
2.2 VNF under test and Metric collection points ............................................................ 4
2.3 Workflow 1: The Developer to Network Service Provider Direct Route .................... 6
2.4 Workflow 2: The Developer to 3rd Party V&V Provider to Network Service Provider .... 7
2.5 Workflow 3: The Developer who sources components ................................................. 8
2.6 Monitored interfaces and components in V&V .......................................................... 12
2.7 Overview of internal components and external interfaces ......................................... 14
2.8 V&V test generation lifecycle ..................................................................................... 20
2.9 V&V test lifecycle ....................................................................................................... 21
2.10 V&V Testing cycle and test generation cycle interaction ............................................ 23
2.11 General Certification Workflow ................................................................................ 24
2.12 Model Based Testing Generic workflow ................................................................... 25
2.13 Black Box test ........................................................................................................... 25
2.14 ETSI NFV reference architecture and Ve-Vfnm reference point ............................... 26
2.15 NS White-box testing .................................................................................................. 26
2.16 TTCN-3 Architecture ............................................................................................... 27
3.1 Test plan, test profile, test suite and test case ............................................................ 33
4.1 5GTANGO monitoring architecture ........................................................................... 39
5.1 V&V and test platform deployment flavors indicating different administrative domains 41
6.1 Metadata lifecycle ...................................................................................................... 46
6.2 Decision Support ........................................................................................................ 46
6.3 Continuous Optimisation ........................................................................................... 47
6.4 Policies Management framework ............................................................................... 49
6.5 SLA Management Framework ................................................................................... 51
1 Introduction

5G will not only be an evolution of mobile broadband networks as its predecessors, but also open the door to unique service capabilities [19]. In this context, Software Defined Networks (SDN) and Network Function Virtualization (NFV) are technologies that will shape the evolution of the telecom sector with new network capabilities and business opportunities. The goal is to increase the level of programmability, control and flexibility of networks, while reducing network operation costs. This will bring more service agility, reducing the time to market of services, and more cost-effective service delivery. These features will empower innovation and creation of new emerging applications that 5G promises. This will also cause disruptive shifts for all the actors in the classic telecommunications value chain, who will have to adapt their strategy and business models to the new paradigm.

Taking into account the current predictions for the adoption of NFV and SDN, it can be observed that, although the technologies are already there, the adoption has only recently started to pick up speed [28]. One of the most promising enterprise NFV use cases is DevOps automation. It is in view of this that we propose an extended NFV architecture that supports DevOps, which requires a re-tooling of the organisation and not only just a technological adoption, as it implies an organisation change in several departments.

One of the biggest challenges in DevOps environments is the validation and verification (V&V) of VNFs and 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. Such a V&V process does not only include functional testing of VNFs and NSs but also non-functional tests, such as performance measurements for gaining insights about resource requirements to fulfil SLAs and to provide the expected Quality of Experience (QoE). To fit seamlessly into the anticipated DevOps workflow, all these V&V procedures need to be fully automated and be able to qualify any VNF or NS without further human interaction.

In this document, we will present our strategy to achieve such DevOps environment enabled by the V&V and other components in the 5GTANGO framework, such as the service platform or the SDK. It will try to answer the following questions:

- What do we want to validate and verify?
- What are the needed functionalities to achieve such goal?
- How do we anticipate to implement them? With which methodology and tools?
- Who are the stakeholders who will use the V&V platform? How?
- What are the needed data and how do we manage them?

By answering the above questions, we provide a clear guideline to implement the necessary 5GTANGO V&V to support a DevOps environment.
2 Overall Verification and Validation Strategy

This section will give an overview of the verification and validation strategy that we want to propose in the 5GTANGO project.

2.1 Ecosystem Workflow

5GTANGO imagines a future ecosystem with various stakeholders taking on roles in validating and verifying network services. It is foreseen that the developer would want to ensure the quality of their own code prior to release, and the network service provider would want to validate and verify all code prior to deployment on their network. Finally it is expected that a range of third party verification and validation organisations would exist that would amortise the cost of verification and validation for independent network service and VNF developers who hope to supply many network service providers. These same third party verification and validation providers who offer network services and other developers with library of pre-verified and validated solutions. With this in mind 5GTANGO is being designed with three different V&V deployment scenarios:

- A developer, in his/her own laptop;
- A 3rd party test organisation, with a relatively sophisticated test infrastructure;
- A network operator, wanting to verify and validate VNFs and NSs bought from several vendors and to be instantiated in its infrastructure.

Different types of configuration support these deployment scenarios. Before progressing to describe the configurations and organisations they support, it is important to understand what we are testing.

2.1.1 What does the Verification and Validation (V&V) Test and Why

2.1.1.1 Why Verification and Validation

The goal of the V&V is to address the concerns that service operators have in hosting 3rd party code upon their network. To address this, the V&V provides a Verification and Validation process which allows the service operator to have some level of confidence in the hosted 3rd party code.

2.1.1.2 What is being Validated and Verified

5GTANGO V&V has two test targets which, although inter-related, require different approaches to successfully test:

1. Network Services;
2. VNFs.
2.1.1.3 Verification and Validation of Network Services

A Network Service is a collection of one or more connected VNFs along with a network service management framework which manages the VNFs in such a way as to allow the Network Service to cope with changes in its environment, while still delivering the network service functionality. The orchestration environment may provide its own service manager, or the network service developer may provide a Service Specific Manager (SSM). Regardless of the source of this management framework it is important that the network service is tested together with whatever management framework will be utilised in the production environment. This is an important step in understanding how the network service will operate and behave under stress. For the purposes of clarity, from here on this management framework, regardless of source, shall be referred to as the “Network Service Manager”, or NSM.

In this scenario, the V&V framework will verify and validate the entire network service including the behaviour of the policy handler. The system under test (SUT) is therefore:

- The collection of VNFs;
- The NSM;
- The Forwarding Graph:

To support the V&V tests the system will collect metric information from each of the VNFs which compose the NS, the NSM and information exchanged between the various VNFs and the NSM.
2.1.1.4 Verification and Validation of VNF

The 5GTANGO V&V will provide developers with a library of VNFs that they can use to create their own network service.

When performing the verification and validation of a VNF the level of testing has to be different. All the interface points for the VNF have to be tested, including how the VNF interfaces with their management frameworks. Orchestration environments provide basic VNFM which provide support for default behavior of the VNF, it is therefore important to test to ensure that the VNF responds to the VNF-VNFM interface. Without this, it would not be possible for a developer to know that they could safely use the VNF as a component within their NS.

Additionally VNFs can be supplied with a FSM (Function Specific Manager) which provides bespoke management of a given VNF. If a VNF is supplied with a FSM then to understand the VNF it is important that the VNF and the FSM are tested together. It is possible that when a NS developer composes a NS of multiple VNFs that they would wish to supply a SSM which could alter the management behavior of VNFs. Therefore regardless of the source of the VNFs management component the VNF must be tested to ensure it can respond appropriately to the managers instructions.

The system under test (SUT) is therefore:

- A single VNF;
- Data in / Data out of the VNF and the VNF’s reported functionality;
- The VNF’s interaction with a management framework;
- If supplied the FSM (optional).

The approach for testing an individual VNF is to automatically package the VNF as a Network Service within the V&V platform. VNFM can be provided by the service platform to interoperate with generic VNFs. It is important to validate and verify that an individual VNF can cooperate with a VNFM.

The V&V framework creates a test-harness network service for each single VNF, complete with an Ancillary Policy Management Driver. The purpose of the ancillary driver is to provide a subset...
of policy functionality with no corresponding management function. This ancillary management
driver is connected to and driven by the V&V Test Engine and the testing currently being executed.

To support the tests which will run to perform the verification and validation, the system will
collect metric information from the VNFs and presupplied ancillary management driver.

2.1.1.5 Verification and Validation Results

Debugging issues with a VNF or a NS requires a plethora of information from low level metrics,
logging, system reporting, system load when the issue occurred, captured input and output inform-
ation. Then all of this data needs to be presented in an easy to use human readable format. It
quite simply is a huge task, one worthy of detailed analysis in it’s own right. It is beyond the scope
of 5GTANGO.

The V&V is not intended as a debugging solution for developers but the results can be used to
support a developer in RCA (Root Cause Analysis) scenarios. It is a verification and validation
platform. It is expected that when a submitted network service fails verification and validation that
a brief report is provided. Full details and root cause of the failure will be down to the developer
to understand via their own development and debugging environment. It is sufficient for the V&V
to provide the failure and the justification of the lack of Verification and Validation.

It is envisioned that the developer may need to recreate a test scenario in order to better un-
derstand the root cause of the failure and to resolve it. The V&V will support the replicability of
tests by recording generated data and system configuration information, and allowing this to be
“played back”. In this way, a developer can recreate a test in their own environment for debugging
purposes.

2.1.2 The V&V and Supporting a V&V Ecosystem

It is intended that the V&V will support the setup and maintenance of a corresponding V&V
ecosystem for the supply of validated results to interested parties. This is similar in concept to
the testing conducted by Apple and Google on submitted applications before they are released to
the Apple Store, and Google Play stores respectively. The exact criteria and the exact level of
confidence differs between Apple and Google, similarly it is anticipated that the exact criteria and
level of confidence will differ between service operators.

2.1.2.1 A Historical Example of 3rd Party Validation

Validating and Verifying network services is a hard and potentially onerous challenge. In the early
2000’s, Nokia and Symbian needed to validate applications which executed on their mobile system.
Rather than testing the applications themselves, Nokia and Symbian outlined the requirements of
the applications to be tested, and allowed third parties to act as testing / validation houses. These
third parties where able to sign and validate code submitted to them. Only signed code could
execute on the Symbian based mobile devices.

2.1.2.2 3rd Party Validation and the V&V Ecosystem

In a similar way, 5GTANGO anticipates that a network of V&V providers will exist. They may
share Validation and Verification criteria, or there may, as in the case of the Nokia example given
above, be an implicit trust between providers.

To enable such an environment, it is anticipated that various stake holders will host V&V envi-
ronments. These stakeholders are:
Figure 2.3: Workflow 1: The Developer to Network Service Provider Direct Route

- The developer, who needs to run simple verification in house before releasing software;
- Third parties, who may act as a trusted source of Validation and Verification for one or more service operators;
- Service Operators, who will want to run their own Validation and Verification processes.

2.1.2.3 V&V Workflows

Given the existence of the 3 stakeholders listed above, it is conceivable that developers of new network services will have to navigate work flows between their own in house testing, 3rd party test providers, and the network service providers. The following is a non-exhaustive list of the types of workflows which might arise:

1. **The Developer to Network Service Provider Direct Route.** The developer has a close working relationship with the Network Service Provider. Perhaps the Network Service Provider has shared some detail on the tests that their V&V runs with the developer. In this scenario the developer creates the new network service and initially validates it in house with his/her own tests. Once the developer is happy with their network service then it is submitted to the network service provider. The network service provider upon receiving a new network service will run its own internal Verification and Validation process, and return the results to the provider. Once the network service provider is happy with the verification and validation process it will add the network service to its internal catalogue, ready for deployment into a network.
2. **The Developer to 3rd Party V&V Provider to Network Service Provider.** As with the previous workflow, the developer creates her/his new network service and validates it in house before sending it to a 3rd Party V&V provider. In this scenario there may be many network service providers who trust the rigorous testing procedure employed by this 3rd party V&V provider. The developer hopes to get access to a much wider set of potential customers when submitting to the 3rd party provider. Once the third party provider receives the network service, it passes the NS through its own set of verification and validation processes. It shares the results of this process with the developer, and if the network service passes the 3rd parties V&V process, it is listed in the 3rd parties catalogue.

Network service providers may source network services from the third party V&V provider. It is expected that they will still want to “double check” and will run these pre-verified network services via their own internal V&V processes.

3. **The Developer who sources components** Developers of network services will want to compose them of VNFs that have already been validated. It is assumed that the smallest network service is a network service consisting of a single VNF. In such a situation a developer
who needs to create a new network service can do it by composing their service of VNFs which have already been verified. Third party V&V providers can provide this functionality, and offer developers pre-approved components from which they can construct their own network service.

2.2 V&V Architecture

2.2.1 Purpose of this Section

This section of the document provides an overview and introduction of the V&V architecture. It provides a description of the framework which can automate the execution of the tests. In doing so, it describes a platform which can support a large spectrum of different tests over multiple configurable testing infrastructures. As described in further detail below, this is achieved by allowing the tests, the NS to be tested, and the test environment be configurable.

The V&V platform is a single component in the 5GTANGO project. It is intended to work seamlessly with the rest of the 5GTANGO software. However to aid software reuse and to offer the
chance of the V&V platform being used with other orchestration environment, the inter-component dependences are kept to a minimum.

2.2.2 V&V in the Global 5GTANGO Architecture

As noted above, the V&V platform is a single component, designed to have limited inter-dependences with other components from within 5GTANGO. The V&V provides testing environment(s) and tooling (e.g. probes, sinks and generators), executes tests and provides signed results. This also includes the sandbox environment available to the developers for testing during development time (not verification).

The V&V is designed to be easily integrated into 5GTANGO. Below there is a brief overview of the different 5GTANGO components that the V&V platform will need to easily integrate with:

- Catalogue: Stores NS + meta data + results, may offer additional functionality (e.g., decision support);
- SDK: Used to develop and annotate NS with some metadata; it can use as a local V&V instance for testing;
- Service platform: Deploys and orchestrates NS that were previously tested by V&V.

2.2.3 Workflow and Results Handling

As mentioned above, the Catalogue and other components help store and manage the results of V&V testing. However following the Unix philosophy of simple tools which DOTADIW[6](Do One Thing and Do It Well) the V&V runs and maintains test state of submitted VNF / NS. The results produced by V&V are stored locally. They are not automatically uploaded or shared with any other system, or another V&V library, rather they require a pull mechanism on behalf of the client system.

The sharing of results is left to the V&V users discretion. In all likelihood automated sharing of results would require the creation of integration scripts / code / components. This is out of the scope of the V&V and will not be addressed in the V&V design, with the exception of the V&V providing support for such integration.

2.2.4 The Three V&V Configurations

To support the workflows described above the V&V needs to be configured for three different types of user:

1. The Developer;
2. The 3rd Party V&V Provider;
3. The Network Service Provider.

The functional and non-functional requirements for each user configuration are different and are described briefly below.
2.2.4.1 The Developer Configuration

The developer will require an easy way to configure V&V which can work with a developer supplied test environment. The developer will be interested in getting much more detailed reports from the V&V and potentially will want to use their debug tools with the V&V as it performs tests. Depending on team size, and particularly for smaller development teams, it is unlikely that the developer would want to spend a long time configuring user access control, and would be more interested in fast ways to setup the environment.

The developer will maintain their own set of tests which the V&V will execute. Depending on their relationship with either a 3rd party V&V provider, or a Network Service Provider the developer may be provided with additional, exported tests from the V&V environments of the aforementioned stakeholders.

2.2.4.2 The 3rd Party V&V Provider Configuration

A third party V&V provider would be interested in running the V&V at a vastly greater scale supporting multiple test environments and running tests concurrently in each environment. They would also be interested in ensuring that the V&V applies the correct user access controls when testing, ensuring that only the submitter of a network service can observe that network service results.

The third party V&V provider may host tests it has been given by network operators and may execute these over their catalogue of network services.

It is likely that a third party V&V provider would have access to greater test infrastructure, and therefore the infrastructure available to the third party would allow larger tests to be run, or scenarios not possible to test by the developer.

2.2.4.3 The Network Service Provider Configuration

The network service provider will have similar requirements to the 3rd party provider, specifically around scale. For user management, it is anticipated that the network service provider will have an existing authorization system which the V&V will need to integrate with.

It is imagined that the network service provider will have specific in-house tests which reflect issues they’ve previously encountered on their network.

It is also imagined that the network service provider will be able to supply a test environment which more accurately reflects to reality of deployment onto their production systems than any other provider can supply.

2.2.4.4 V&V Configuration

In each configuration, the following must change:

1. **The Test Infrastructure Environment**: Each test environment will have a different test infrastructure. This will be dictated by the platform, but the architecture should accommodate this. In the current standards models and industry approaches, this is done not via a VIM like interfaces, but via a domain based orchestration interface. For the V&V, this means that it needs to communicate with the orchestration layer for NS deployment. This interface will need to be extended to allow for the corresponding metrics to be made available to the V&V. These metrics from the orchestration and infrastructure layer are required to support the types of characteristics tests envisioned. The 5GTANGO project is committed to
contributes to existing open source projects, this implies different orchestration platforms, and hence orchestration interfaces.

2. **The Set of Tests to be executed**: The set of tests will change, the developer may have a set developed in house, and the 3rd party V&V may have a second set which is specific to them, of course each service provider will have their own. Some tests may be shared and some will be offered by the platform catalogue. This needs to be accommodated by the V&V architecture.

3. **The user access to the V&V**: When deployed as a developer tool, user access control to the V&V can be straight forward; however with limited control. But if the V&V is deployed as a 3rd party tool, or inside a network operator, it will become increasingly important to ensure that developers of competing NS are isolated and constrained to data local to their access privileges. As a 5GTANGO component, the V&V should share the same user access control as any other component within 5GTANGO.

### 2.2.5 Architecture Description

#### 2.2.5.1 External Overview

To interact with a V&V platform, a simple interface is provided that can be used by any role or component in a similar fashion. Specifically, developers, catalogues, operators, etc. use the same interface to interact with a V&V platform.

Through the external interface, users can provide **input** to the V&V. It allows uploading a NS and metadata for testing and for triggering a test. Tests do not need to be triggered immediately. Instead, they can be triggered manually at any given time or automatically, e.g., by changes in the test configuration or be queued and executed in sequence in a FIFO mode. Note that the V&V platform will be capable of running test jobs in parallel subject to the design of the test suite and any associated resource limitations in the underlying target platform provided by the V&V store.

Furthermore, the interface allows the **configuration** of a V&V platform. Both tests and infrastructure can be configured.

The V&V provides **output** such as test results and testing environment metadata through the interface. Specifically the V&V:

- Publishes notifications about the current test state: queued / pending, executing, completed;
- Allows enumeration of all available test results;
- Provides access to summary information about test results (PASSED / FAILED);
- Provides access to detailed information about a test result (summary, and observed / recorded metrics);
- Allows downloading of the test results in a digitally signed package.

Access to the results is controlled, ensuring that only users who have the relevant permissions can access the results. This ensures that user A can’t see the results from a network service submitted by user B.
2.2.5.2 Internal Overview

Internally, the V&V uses separate repositories to manage submitted network services, tests, and test results. New NSs are submitted to the NS catalogue. Once the client triggers tests (or they are triggered automatically), the corresponding NS and suitable test are selected and loaded from the repositories.

The test engine performs the selected tests on the network service using the service platform. During test execution, several metrics are continuously monitored and logged (as configured by the developer or from the policies). The test engine issues events signaling the changing state of a test, from pending, to executing, to complete.

When a test is complete the test engine will:

1. Ensure that the test results and associated recorded metrics are stored in the result repository.
2. Issue an event signaling that the test is complete.

External V&V users who have registered for notification of when a test is complete will receive this notification. External users can download the results via the API Gateway and the Results Repository, as noted above in Sec. 2.2.5.1.

2.2.5.3 Monitoring

Fig. 2.6 provides a description of the interfaces and components that are monitored during test execution.

Repeating Tests

The following must be recorded in order to ensure that tests can be repeated:
• **Input**: the data generated which is passed through the NS is recorded. This allows the exact same set of data to be “played back” when the test is repeated.

• **Output**: To check for functional consistency, the NS produced data is also recorded. This allows test to be repeated and the functionality between test runs to be compared.

### Metric Collection

A monitoring system collects metrics from various components within the network service, these include:

- Metrics from the physical infrastructure on the stress imposed by the network service;
- Metrics from the virtual infrastructure on the stress imposed by the network service on the virtual infrastructure;
- Published metrics from the VNF on how it is operating;
- Published metrics from the NSM on how it is operating (SSM if supplied, or default manager);
- Published metrics from the FSM on how it is operating (FSM if supplied, and / or default manager);
- Instructions and API interaction between the NSM, FSM and the VNF.

To gather all this information it may be necessary for the monitoring system to deploy probes, or to activate physical / virtual metric collection systems. All of this can be configured by the tests before they execute. This functionality is available via the Test Platform Manager (see below) which provides an API for managing the test platform and the metrics collected.

### 2.2.5.4 The Test State Machine

As discussed above, and echoed in D2.1, there are a number of requirements around the automatic execution of tests when either:

- The test environment configuration changes;
- The test and / or its associated metadata changes;
- The NS and / or it’s associated metadata changes.

The V&V must maintain state information about the tests which have been updated / executed against which NS, as this information is required to correctly schedule new tests when any of the information listed above is modified.

The V&V will issue a new test execution request when any of the data list above is modified. It will need to maintain a list of the NS and all the tests they have had executed against them.

### 2.2.6 Architecture and Component Details

The following diagram provides an overview of the major internal components of the V&V and the external interfaces (marked as API) they provide.

Each of the components in the diagram above are described, below:
Figure 2.7: Overview of internal components and external interfaces
2.2.6.1 Component Description

Based on the elaboration of the initial requirements identified in D2.1 and covered above, we can identify the following components. In the rest of the section, all the requirements represented as VVxx are described in D2.1.

User Management and Public API End Point

- **“AAA”**: The V&V does not implement its own user management solution, instead it provides as a plugin architecture allowing an existing user management solution to be supplied. Every component which provides a functional interface verifies the caller using this plugin solution. This is done at both the external interface location (V&V API Gateway) and at the major internal providers. This ensures that even if an attacker were to breach the external boundary of the service attempts to invoke the APIs would still require the same level of authentication. For instance, SONATA[1] contains its own User Management System[27], the V&V will support a configuration option which will allow it to integrate with and support the centralised user management offered by the SONATA Gatekeeper component. While this check at every interface adds some additional overhead for the initial implementation it supports the great flexibility of implementation required by 5GTANGO. This supports requirement VV01, VV17, and VV32, VV33

- **V&V API Gateway**: The API gateway provides a common entry point for all V&V API calls. It acts as a dispatcher, routing invocations to the correct API provider. User authentication is performed here.

Internal Repositories

- **Test Repository**: The test repository will allow tests and their associated metadata to be managed. This will address requirements: VV04, VV05, VV06, VV07, VV11, VV18, VV19, VV20, VV21 and VV42. The Repository will generate events (notifications) of when a test is changed. These change events can be used to support triggering the re-running of tests on already stored NS, see requirement VV13. See sec. 3.1 for more details.

- **Package Validator Test(s)**: For every supported platform, the test library will have as a minimum a test to ensure that a supplied network service contains a valid package. The V&V’s own package format is a container structure, similar in concept to the MP4 specification, which defines the file format, not the codec being used. The V&V package format supports multiple orchestration packages in the same network service package. This would allow a network service with multiple implementations, one for each network service to be created. The Package Validator tests are orchestration specific package validation tests.

- **NS Catalogue**: For the V&V to be able to schedule and execute tests, it must know from where to obtain the NS for testing, and the state of the NS (modified etc). To fulfill this the V&V supplies a NS library which stores to the NS ready for deployment in the NS environment. Supporting VV23, this component provides the external API for uploading and managing NS to be tested by the V&V, as such it performs the initial package validation:

- **Package Validator**: This ensures that the package uploaded confirms to the V&V container specification. It does not do a per orchestrator package check, rather it confirms that the package adheres to the V&V specification.
• **Test Results Repository**: Once a test has completed execution, the results need to be available to allow a test to be re-executed with the same input data / conditions, or retrieved by third parties. To support this the V&V has a Test Results Library which exposed an API providing access to the results from test execution. This helps achieve VV24, VV25, VV28. It provides a set of APIs for managing the test results, including a CRUD (Create-Retrieve-Update-Delete) like interface. For more detailed information on the Test Results Repository please see sec. 3.2. Notes that all test results are annotated with information detailing the infrastructure upon which they were executed. See sec. 5 for more information.

• **Test Result Analyzer**: This receives the raw data from the test engine as the tests run, and uses it to generate the overall test result (star system).

### Managing State and (Automatic) Invocation of Tests

The requirements document D2.1[7] requires that the test engine monitors the state of all of the network services it knows about, and the tests it has been given, along with their infrastructure configuration. If anything was to change, the network service updated, the tests change, or even if the metadata about either of these things change, then the V&V must calculate which new tests it needs to execute against which network services in order to ensure that all the network services it knows about are validated and verified. This task is taken on by the Test Invoker.

• **Test Invoker**: It is necessary to watch for events which can trigger automated test execution. These are state changes are listed above, namely changes to test environment configuration, test metadata, test, NS, or NS metadata. The NS and Library Watcher receives the notifications of changes and using the available information generates the list of tests which need to be executed and passes that on to the Test Engine; this supports requirement VV13.

• **Test Scheduler**: This works with the test engine and will schedule new tests if they are required. It can remove and replace tests already on the queue if the changes the Invoker has been made aware of require it.

• **Test Configuration**: Every test needs to be configured with the necessary data to use as input and expected output, along with deployment configuration options. The test configuration component is aware of this and can ensure that test, are scheduled with the most effective configuration.

• **Test State Manager**: This records and manages the state of all the tests and the network services that have been tested. This ensures that double testing doesn’t occur, and only the necessary tests to ensure completed configuration are scheduled. In addition, using the Test Configuration, this can ensure that expect tests are repeated, along with replica input data.

• **The Test Engine**: The Test Engine is a network service testing specific workflow engine. It consumes test requests from the Queue of Tests and then executes these. The test engine is responsible for executing tests, and controlling the queue of tests due to be executed. This allows it to manage waiting list of tests to be executed and provide an API allow for new tests to be queued, queries on the state of a test (complete, pending, executing, etc.) and it also allows the Test Engine to generate its own events for when tests are being executed or queued (queued, executing, and complete). This supports requirements VV10, VV11, VV14, VV16, VV15.

When the tests are complete, the Test Engine must store the results in the Test Results Library. The workflow follows the same basic structure:
• **Platform Specific Validation**: Prior to deploying the network service, its package is tested to ensure that it could be deployed. This is done using Package Validator Test listed in the Test Library Manager.

• **NS Platform Deployment**: Once the package test has passed the network service can be deployed, the NS Platform Deployment component performs this task by interfacing with the logical interface presented by the Test Platform Manager. This can be considered the set up necessary for a test plan, see sec. 3.1 for more information.

• **Test Technology Adapter**: The main body of the workflow manager. This element obtains the tests and their configuration and then executes them. It supports various types of test technology plugins including, Bash Scripts, and TTCN-3 scripts.

• **The NS Platform Decommissioning**: After executing the suite of required tests the NS needs to be decommissioned. This is handled by the Test Engine using this component.

**Interfacing with multiple Test Platforms**

As discussed in sec. 5 there are main different types of test environment that may be configured for use with a V&V. Each of these test environments can be considered an orchestration domain. The main point of interface for the V&V is therefore to the domain specific orchestrator. To allow multiple domain orchestrators to be used, the V&V will define an pluggable interface to the domain orchestrator. In doing so it provides layers of abstraction allowing the tests, and the test engine to see a high level abstract interface. The specifics of the particular binding to an orchestration framework are abstracted away to lower layers.

The test environment needs to do more that just execute a network service, it also needs to supply metrics from the orchestration layer, virtual and physical layers. So support this a monitoring system is provided. This too is implemented using the same logical approach, with the Test Platform Manager supplying a high level interface, and allowing a pluggable lower layer to support the specific implementation.

• **Test Platform Manager**: This provides the test engine with a high level interface for network service deployment. This component contains a lot of the logic required to actually deploy, run, and tear down network services. It also provides a high level interface to allow the collection of advanced metrics. This high level logical interface is implemented using two sub components:
  - **NS Lifecycle Manager**: This is component is responsible for understanding the logical steps in deploying and managing a network service. This can be considered a “logical driver” as it contains the logic, but not the specific implementation. The specific implementation is performed by the Orchestration Driver Framework, described below.
  - **Metrics and Monitoring**: This is also a logical driver, containing the logic to collect together the metric information, and to control the flow of this information, turning probes on / off etc. However the probes and their implementation are performed by components in the Orchestration Driver Framework, described below.

• **Orchestration Driver Framework**: This is the actual “physical” driver. Here a set of plugins exist, which all implement the same core API but actually talk to and interface with the orchestration frameworks. Plugins for 5GTANGO, OSM and ONAP are expected. At the same time, this component also exposes a set of orchestration specific metric and monitoring systems. These monitoring solutions are provided for the corresponding 5GTANGO, OSM and ONAP orchestrator environments.
2.2.7 Supported Scenarios

The architecture described here supports a number of advanced scenarios. These go beyond what is typically provided by test frameworks and are tailored to addressing the dynamic nature of the test environment.

2.2.7.1 Supporting the Dynamic Test Environment

In a software service environment, tests are never static; the set of tests changes in production and new test cases are created. In the world of a network service provider, once an issue is resolved with one network service, it will be necessary to ensure that all other network services are resilient against the same fault.

The automated invocation of tests outlined here addresses this. When a new test is added to the test repository it will automatically be applied to all network services and VNFs which match the tests metadata. This way the network operator can maintain a list of known good network services and VNFs, and can quickly identify software which has issues.

This approach has some interesting side effects when a new test, or a change to an exist test occurs. When the developer loads this to the test repository, the Test Invoker is informed of the change and it will automatically schedule tests for execution. Similarly, when a network service is updated, it will automatically be scheduled for verification and validation.

2.2.7.2 Supporting Multiple Test Environments

The V&V can support more than one type of orchestration environment. Assume a developer uploads a 5GTANGO network service package which contains implementations of a network service for ONAP and 5GTANGO. Assuming that the V&V engine has access to these two test environments via the Test Platform Manager, it will execute the tests against the two environments. This allows a developer to qualify a network service against multiple environments with ease.

Likewise, and perhaps more usefully, a single VNF could be qualified against multiple orchestration environments. When a VNF is submitted for testing, the V&V framework automatically wraps it as a Network Service and deploys it to the test environment. Such an approach would allow a VNF to be wrapped in multiple Network Services, one for each orchestration environment, then tested as a multiply packaged Network Service (described above).

2.2.7.3 Developer Test Replicability

As a developer, there is nothing more frustrating than being told that your software did not pass a test, and not being given any further information as to why. To help the developer overcome this situation, the V&V can be deployed on the developer’s own environment. Not only that, but by exporting the tests from a network operator’s V&V platform and importing it into the developers environment, it allows the developer to run the very same tests in their own environment.

This is useful, and yet the V&V goes further. By recording the network service or VNF’s infrastructure configuration, and the input and output data of a test pass, it enables the exact same test to be replicated by the developer.

In this way, the developer can easily reproduce verification and validation failures, which helps significantly in resolving issues.
2.2.7.4 Unique Test Environments

The ability of the developer to have their own V&V framework means that they can create infrastructure environments which are specific to their own use cases, adding physical network functions or other such attributes.

2.2.7.5 Default Tests

It is anticipated that the V&V will come with a set of default test scenarios. As mentioned above, the exact tests are the subject of a future document.

2.3 V&V Methodology

The objective of the 5GTANGO V&V platform is to provide a modular and vendor-independent facility for automatically test and validate VNFs or NSs submitted to the 5GTANGO Catalogue. Together with the Service Development Kit (SDK), the Catalogue and the Service Platform, the 5GTANGO ecosystem aims at providing an NFV DevOps model to accelerate the new NFV services development and deployment.

The V&V platform will be firstly made available for 5GTANGO platform. However, it will be designed to be generic and easily adaptable to all NFV-compliant infrastructure, it will be easily replicated in any operator’s NFV infrastructure, or even multiple-operator infrastructure. In 5GTANGO, as explained in the architecture section, more than one NFV infrastructures may co-exist, being it public or not. It will be possible to replicate one or more target operator’s infrastructure including co-existance of public/private configurations

In this section, we will explain the general 5GTANGO DevOps lifecycle, the concepts and the tools that will be applied to achieve such DevOps model.

2.3.1 V&V Lifecycle

This subsection presents the lifecycle of the V&V operations that refer to all the features and activities that a V&V platform is capable of realising, such VNF/NS testing, test generation or metadata generation.

In the V&V platform, we can distinguish two subjects which need lifecycle management:

1. **Test generation**: As the name suggests, it manages to create and update tests that will be run upon the VNFs or NSs to be validated and verified;

2. **VNF/NS testing**: It consists of the automatic testing and metadata annotation on the VNF/NS.

These will be treated in the following sub-sections.

2.3.1.1 Test Generation Lifecycle

Fig. 2.8 shows an overview of the test generation lifecycle. The input of this lifecycle are requirements or subjects that need to be tested, and the output is a set of executable tests ready to be run on the VNF/NS.

1. **New/Updated Requirements**: This input comes from the 5GTANGO Service Platform’s requirements as described in [7], from the standard specifications (more details provided in
sec. 2.3.6) and even from operator infrastructure’s specific requirements if the V&V platform instance is deployed in a specific infrastructure;

2. **Development of an Abstract Test Model:** Abstract Test Model is a high-level model derived from the model of SUT (System Under Test). It describes the test objectives, and the abstract test behaviour. But it cannot be used directly upon the SUT because of lack of concrete details of the SUT for execution. The abstract test model has two advantages: 1) provides a high-level common understanding about the tests between the test developers and the test users; 2) enables the automatic (executable) test generation in some of the MBT (Model-Based Testing) tools. More details about abstract test model and the test generation will be provided in sec. 2.3.3 MBT;

3. **Test validation:** In order to make sure that the developed tests do what the input requirements ask to do, validation on these tests are necessary. The validation is usually a manual debugging process which crosses several SUT implementations and test implementations to verify if the results from different test implementations are consistent upon the same SUT, and if different SUT implementations obtain their propre results;

4. **Store the tests in a test repository:** In the test repository, abstract tests as explained in step 2 are stored and annotated that facilitate the test users to research and retrieve useful tests to run on his/her SUT. More details of the test repository and categorisation will be explained in sec. 3.1.

### 2.3.1.2 VNF/NS Testing Lifecycle

One user of 5GTANGO’s V&V platform is the **service developer** as described in sec. 2.2.4. The service developer creates or reuses available VNFs in order to produce a new NS. Thus, the testing consists of two levels: the testing of new VNFs (if they are created for this specific NS, i.e., are not reused) and the testing of the whole NS which, comprises one or more of those VNFs. The lifecycle of a VNF is managed by a specific manager, where as the lifecycle of an NS consists of VNF managers or modular function-specific plugins plus their own service orchestrator components (or service specific management plugins), packaged along with descriptors for the deployment and configuration of VNFs. In summary, the lifecycle of service validation and verification is illustrated in fig. 2.9.

1. **Local testing.** The service developer can run the tests in his/her local V&V environment such as in the emulated environment delivered together with the SDK;
Figure 2.9: V&V test lifecycle
2. **Submit to an infrastructural V&V.** Once the developer is satisfied with the local testing results, he/she can submit the VNF/NS to the 5GTANGO platform together with the service description. Based on the service description and the required additional information on tests, a set of tests is selected to be executed. The selection of tests will be explained in Test suites and test category section sec. 3.1;

3. **Test execution in test platform.** The selected abstract tests are made executable with the concrete information from the VNF/NS and the test execution platform. They are executed by the test engine in upon a test platform which is a replication of the 5GTANGO service platform with configurable parameters. More details about the test platform are provided in sec. 5;

4. **Test result collection and metadata annotation.** The test results are collected. Together with a set of metadata such as the test environment configuration, they are packaged and bundled to the SUT, and submitted to the Catalogue;

5. **VNF/NS storage in Catalogue.** The Catalogue decides to store or not the VNF/NS submitted regarding to the test results. The decision process is out of the scope of the V&V platform. The stored VNF/NS is public or private regarding to the choice of the developer marked in the very initial metadata from step 2;

6. **VNF/NS deployment in service platform.** The Service Provider request to instantiate the service and the requested service is deployed in the production environment by choosing from the Catalogue;

7. **Monitoring in service platform and feedback.** Monitoring on the production environment and on the deployed service allows to check continuously whether the service meets the SLA or whether the service encounters problems that were not revealed in the testing environment. If necessary, the deployed service will be suspended temporarily for fix-up. More details of the monitoring functionality are given in sec. 4;

8. **Modification of the VNF/NS.** The developer modifies the VNF based on feedback from the testing cycle or through changes in the overall requirements on the VNF. Once updates are committed, the test cycle can be repeated.

The two lifecycles (Test generation and VNF/NS testing) interact through the test repository where generated abstract tests are stored with metadata to be chosen by the testing cycle (shown in fig. 2.10).

### 2.3.2 Certification and Labelling Process

Certifications (or equivalent processes) are organized all over the world in mostly all industrial domains either for regulators or for organizations on a voluntary basis. The aim of a certification process is to ascertain the conformity which is defined as the *fact that a product, system, body or even a person... meets specified requirements* [5] and which can improve the business interests with regard to products, goods and services.

The 5GTANGO V&V framework aims to deliver such a certificate to the VNF/NS under test if they meet all the requirements, and adopts a certification process approach, which means the V&V considers only whether the requirements are satisfied (*what* should be performed by the VNF/NS), not *how* they are satisfied, which corresponds to a black-box approach.
Fig. 2.11 shows a general certification workflow. The certification authority collects requirements and specifications as subject of certification, and defines the tools and processes to carry out the certification. The certification platform implements the tests to check the conformity with respect to the defined tools and processes. Once an applicant submits his/her product (i.e. a device, a piece of software, a service), a set of tests are executed on the product in a controlled and trusted environment. The test result is fed back to the certification platform which declares the conformity of the product if the result is satisfactory. A certification can then be issued by the authority towards the applicant.

The proposed V&V framework can be easily mapped to the above workflow. A group of stakeholders, including infrastructure providers and operators, is responsible for defining the specifications and requirements that need to be met by VNFs/NSs. The V&V platform provider develops test cases and test suites, possibly with contribution from developers and other stakeholders, and provides a reference testing environment for test execution. Test results are collected and transferred to the Catalogue with basic processing, including transformation to a specific format and attachment of digital signatures. If the results are satisfactory to conclude the conformity of the VNF/NS, the VNF/NS gets a digitally signed metadata set serving as certification, and should be made available for deployment by the Catalogue.

2.3.3 Model-based Testing for Automatic Test Generation

Model-based testing (MBT) [30] is a variant of testing that relies on explicit behaviour models that encode the intended behaviours of a SUT and/or the behaviour of its environment. Test cases are generated from one of these models or their combination, and then executed on the SUT. It is considered to be a lightweight formal method to validate software systems. It’s formal because it works out of a formal (that is, machine-readable) specification (or model) of the software system one intends to test (a.k.a. SUT). It’s lightweight because, contrary to other formal methods, MBT doesn’t aim at mathematically proving an implementation which matches the specifications under all possible circumstances. What MBT does is to systematically generate from the model a collection of tests (a test suite) that, when running against the SUT, will provide sufficient confidence that it behaves as the model predicted it would.
Figure 2.11: General Certification Workflow

The fundamental MBT process includes activities such as test planning and controls, test analysis and design (which includes MBT modelling, choosing suitable test selection criteria), test generation, test implementation and execution. Among all the advantages that MBT offers, we can list here several:

- The automatic derivation of abstract test cases from abstract models. Abstract model and test are re-usable for similar systems which can be modelled in the same way. It reduces enormously the test developer’s work load and makes the model evolution automatically propagated to all the generated tests. Executable tests can also be generated automatically with provided concrete system information;

- The traceability between requirements/specification and test cases is maintained. The modelling language allows to precise about the requirements covered for each of the test cases. The tester will have a clear view of which requirements/specification are covered and satisfied when he/she gets the results of the tests.

The assumption here is that models are specified with languages that are sufficiently precise to allow, in principle, a machine to derive tests from these models.

There is a list of MBT tools on the market, commercial, experimental or open source. We cite here only several stable commercial ones: Pragmadev[26], CertifyIt[3], TestCast-MBT[2]. All of the tools provide the MBT features with more or less assistant and utility tools and interface to the software editor’s other products.

Fig. 2.12 shows the generic workflow of MBT.

From the SUT specification document, test analyst (someone who is specialised in test model abstraction, or someone who knows well the SUT) writes the MBT models and the system’s dynamical behaviour in OCL (Object Constraint Language, a domain specific language that describes the dynamic behaviour of the model). Abstract tests are generated from the model and the behaviour description, and stored in a database, which can be a database or simply a set of textual test descriptions. The Mapping file shown in fig. 2.12 is to map the concrete configuration information to the abstract messages in the MBT tests. For example, a postcondition to be checked of a HTTP
API abstract test is to obtain an “OK message” which is to be mapped to “http.code == 200” as a concrete information. The Publisher transcodes the MBT abstract tests to executable tests, either in a script language (i.e. Javascript), or in a classical programming language (i.e. Java, C++). The test scripts are then to be executed upon the SUT by the test engine in a test environment. The test results are collected and stored for further analysis.

### 2.3.4 Black Box vs White Box Testing

**Blackbox testing** (shown in fig. 2.13) refers to examining the SUT regarding its capabilities without the knowledge of its internal structure, which means that given an input following some specification, blackbox testing verifies whether the SUT behaves correctly and emits the expected output. Therefore, the black box testing is most suitable for interface conformance testing as the specification of an interface is exactly a description of expected input and output without considering the implementation that realizes such behavior. For example, in the ETSI NFV reference architecture defines several references points between the MANO and the infrastructure and the NSs. Some of these reference points have been specified as REST API and the specification are available online, such as SOL002[15] for the Ve-Vfnm interface (as in fig. 2.14). The SUT behind the interface is considered as blackbox in the point of view of the tester. As long as the SUT returns the correct output upon a given input regarding to the specification, the tester concludes that the test is successful no matter how the SUT implements this behavior.
In 5GTANGO, specifications and requirements come from standard specifications (e.g., ETSI-NFV), the use cases scenarios, and deployment environments. The perimeter of the SUT needs to be clearly determined (e.g., whether the composed NS is to be tested or just a single VNF) in order to apply appropriate blackbox tests. Functional and non-functional tests are both possible using the blackbox approach.

**White-box testing** consists of testing the internal structure of the SUT. It requires a good knowledge of the internal structure or code of the SUT but it can give more insight of the SUT’s behavior or performance by knowing how the behavior is implemented or how the performance is achieved.

In 5GTANGO, the white-box testing will mostly applied on the Network Service Profiling which refers to testing and analyzing the performance of an NS knowing its internal graph (illustrated in fig. 2.15). We consider a network service (graph) definition that follows a microservices-based approach that can be functionally decomposed into a set of loosely coupled collaborating functions that interact through well-defined interfaces and possibly depend among themselves. The white-box NS graph profiling will take into account of monitoring data on the decomposed functions to analyze and identify how graph relationships affect the overall performance, and where is the bottle neck that deteriorate the performance of the overall graph. It will help NS developer and the orchestrator to optimize the NS graph. More details will be described in sec. 6.1.2.

### 2.3.5 TTCN-3 Test Scripting Language

In the proposed V&V platform, we decide to use TTCN-3[4] for test development, generation and execution. Testing and Test Control Notation version 3 (TTCN-3) is a test scripting language widely
known in the telecommunication sector. It is used by the third Generation Partnership Project (3GPP) for interoperability and certification testing, including the prestigious test suite for Long Term Evolution (LTE)/4G terminals. Also, the European Telecommunication Standards Institute (ETSI), the language’s maintainer, is using it in all of its projects and standards’ initiatives, like oneM2M. A test case produced by TTCN-3 is abstract, because it does not know how or where to send the content, it only knows that it has to send it. This is where the TTCN-3 Control Interface (TCI) and the TTCN-3 Runtime Interface (TRI) get into the process. The TCI is composed of Test Management (TM) module, Test Logging (TL) module, Coding and Decoding (CD/Codec) module and Component Handling (CH) module. The TRI contains System Adapter (SA) and Platform Adapter (PA) modules. All of these modules are already provided by most TTCN-3 tools, except the Codec and the SA modules. The first is needed to convert TTCN-3 structures to a binary, text, XML, JSON or some other serializable format, and the latter is needed in order to communicate with the SUT, because it implements the protocols how the converted data from the codec can be sent or received from the SUT (for example convert the structure to a HTTP request or response, if the protocol used is HTTP).

TTCN-3 test cases can be generated manually or automatically using advanced methods such as MBT which improves the traceability of tests in respect to requirements and easing the maintenance of the repository, at the expenses of strongest skills needed to develop the test model.

2.3.6 ETSI NFV Testing Methodology Alignment

The last couple of years, the ETSI NFV ISG TST working group has produced a number of drafts related to NFV testing methodology and NFV pre-deployment testing. 5GTANGO’s V&V framework falls within the focal area of this group. This section attempts a summary of the main recommendations on validation methodologies and tools that are discussed within the TST WG.

The following documents have been published by the TST WG:

- ETSI GS NFV-TST 001 V1.1.1 (2016-04), Network Functions Virtualisation (NFV): Pre-


- ETSI GS NFV-TST 004 V1.1.1 (2017-05), Network Functions Virtualisation (NFV); Testing; Guidelines for Test Plan on Path Implementation through NGVI - provides guidelines for test plans that assess different approaches to defining SDN Applications, different ways of arranging and federating SDN Controllers, and arrangements of network switching/forwarding functions (both physical and virtual) to create the various path-implementations between and among NS Endpoints and VNFs. The document is mostly focused on the testing of SDN integration with NFV with a few extensions to low-level operation enabling the Service Function Chaining.

- ETSI GS NFV-TST 005 , Network Functions Virtualisation (NFV); Continuous Development and Integration; Report on use cases and recommendations for VNF Snapshot - provides a report on use cases, recommendations and potential solutions for VNF snapshotting.

- ETSI GS NFV-TST 008 V2.1.1 (2017-05), Network Functions Virtualisation (NFV) Release 2; Testing; NFVI Compute and Network Metrics Specification - specifies detailed and vendor-agnostic key operational performance metrics at different layers of the NFVI, especially processor usage and network interface usage metrics. The document will be used in order to align the metrics that the V&V environment will support.

Currently under the group there are work items that are still under discussion without publicly released document, such as:

- ETSI GS NFV-TST 003; Network Functions Virtualisation (NFV); Open Source Components for NFV - aims in creating and maintaining a feedback loop between the ISG NFV and relevant open source projects (for example, OPNFV, OpenStack, OpenDaylight), reducing the “impedance mismatch”

- ETSI GS NFV-TST 006 Network Functions Virtualisation (NFV); Testing; Report on NFV CICD and Devops CICD & Devops report - provides information on how to leverage DevOps and CI/CD techniques across the boundary from SW provider to service provider, or any combination of developer, installation and operational entities. The document includes use cases for the DevOps operation. This document is still at the very early stages and constitutes a possible target for 5GTANGO contributions.

- ETSI GS NFV-TST 007 Network Function Virtualization (NFV); Testing; Guidelines on Interoperability Testing for VIM/VNFM - the document is focusing on features applicable directly for interoperability between VIM and VNF (Vi-Vnfm reference point). This item is also within the scope of V&V framework of 5GTANGO.

- ETSI GS NFV-TST 009, Network Functions Virtualisation (NFV) Testing Specification of Networking Benchmarks and Measurement Methods for NFVI - the document will focus mostly in NFVI testing. It will specify vendor-agnostic definitions of performance metrics
and the associated methods of measurement for Benchmarking networks supported in the NFVI. This work item falls a bit outside of the V&V framework focus area.

Some key points that serve as guidelines for the definition of methodology by 5GTANGO are:

- Currently the use of Devops in NFV environments is also tackled in the TST working group. The status of the currently available inputs and documents reveal that the topic is under discussion and the use of verification and validation as proposed by 5GTANGO could provide important input to form the ETSI NFV view on the topic.

- ETSI validation activities discussed in the documents are targeting all possible SUT or DUT in the MANO architecture, hence the impact area is larger than the one targeted by 5GTANGO. The project addresses specifically Network Service and VNF V&V needs. However as 5GTANGO Service Platform is also allowing the developers to develop code that is part of the VNFM, some tests related to the performance of these components will be also tested.

- There are a number of upstream projects of OPNFV[10] that addresses experimentation and performance evaluation (i.e. Projects Yardstick, VSperf and Vpe Performance Testing). Due to the nature of OPNFV, the projects target mostly on performance evaluation of the NFVI platform and its components. A lot of the available experiments and test cases involved black and white box testing of VNFs under all kinds of load and testing scenarios. In this context and because of the achieved level of automation and the tooling availability, it is considered that 5GTANGO could benefit when entering the implementation stage.
3 V&V Testing Approach

This section will provide more details on the testing approach that 5GTANGO will adopt, including the testing techniques applied, the management of tests, and the management of test results to facilitate their exploitation in related 5GTANGO components such as the Catalogue.

3.1 Tests Management

3.1.1 Test Categories

We can distinguish different types or categories of tests. The following is a non-exhaustive list of categories that we foresee in the V&V platform.

- **Functional testing.** It consists of testing a slice of functionality in a system. The slice of functionality can be a unit of system behaviour, a complex behaviour composed by many unit functions, and also can be the whole system’s behaviour. It aims to test whether the expected behaviour is successfully done by the system rather than how the behaviour is done with which quality;

- **Performance testing.** It is a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload. Performance testing measures the quality attributes of the system, such as scalability, reliability, latency and resource usage;

- **Syntax testing.** It is a static testing which means it does not test the behaviour of the SUT during runtime. It tests the static information associated to the system such as the description file, the metadata. In SONATA project, such tests have been implemented to check the correctness of the VNF descriptor;

- **API testing.** It is part of the functional testing that aims to test the implemented API behaves as expected in the specifications;

- **Security testing.** This testing technique consists of determining if an information system protects data and maintains functionality as intended. It can involve the above testing techniques to test for example: 1) if an authentication functionality is correctly implemented; 2) if the security policy description is correctly written; 3) if the consumption of resource faced to an attack is controlled and isolated; etc.

The goal of categorizing the tests is to facilitate the test suite developer or the V&V provider to efficiently search and find the test they need to test a system comprehensively, as well as to facilitate to define priority of tests if the testing resource is limited.

In addition performance testing category can be further divided into three subcategories based on the which plane/layer the testing takes place. The following subsections are discussing these subcategories (see also ETSI TST001[13] as reference).
3.1.1.1 Data Plane Validation

For these type of validation, standard benchmarking methods are used to perform data plane validation of DUT. Usually applicable to physical devices but will be adapted to be used by 5GTANGO. Some of the most important validation tests are:

- IETF RFC 2544[9] - specifies methods to assess network interconnect devices and measures metrics such as throughput, latency, frame loss rate, and system recovery time;
- IETF RFC 2889[17] - specifies methods for benchmarking of LAN switching devices and takes into consideration flooding and MAC address learning;

Under this regime, test traffic is send by a testing device (source) through the DUT, forwarded by the later and received at another test device (sink). The data received are analysed in order to conclude on the data plane performance. Multiple DUT can also be daisy chained in order to measure end-to-end characteristics of the data plane.

The aforementioned test is applicable to SUT consisting either but single VNFs or VNFCs and also by Network Services comprising of multiple VNFs.

3.1.1.2 Control Plane Validation

Control plane validation and benchmarking methodologies suggest the testing of the DUT through testing devices that speak the same control plane protocols. Any testing of IETF RFC based protocols (e.g. RFC 4271) or any other proprietary or standard control protocol (e.g. OpenFlow) falls under the control plane validation category. The testing process involves test devices that benchmark the DUT by:

- Incrementing the number of control connections to extract the maximum sessions supported by the DUT;
- Varying the session duration (set-up and tear-down) in order to assess DUT performance;
- Verifying the protocol layers (in case of encapsulation or tunnelling);
- Emulating end-points and end-to-end networking (e.g. for testing routing protocols).

3.1.1.3 Management Plane Validation

Management plane tests relate to fault detection, recovery and convergence capabilities of the DUT. Testing of the management plane for a Network Service is basically covered during development epoch. However 5GTANGO provides the capability to the developer to run service specific tests at a sand-boxed environment. The tests and the test devices (in form of VMs) need to be provided by the developer (if not available by the V&V test repository at the first place). Especially for failover and convergence tests, the test probes need to be more than two.

3.1.2 Considerations on Impact of Virtualisation on Testing Methods

NFV facilitates transition from hardware-based network devices, such as firewall, load-balancer etc, to software-based ones as VNFs on commodity servers. VNF implementation styles can be
categorised into virtual machine based (VMs) and container-based. However, any type of virtualisation technology may cause degradation of packet forwarding performance in NFV nodes because of general purpose implementation of packet I/O architectures [16]. In addition, performance might further decrease due to implementation overheads of both packet forwarding implementations of VNFs and virtual network I/O mechanisms that allow networking between a hypervisor and VMs. In addition to the above new components and the new interfaces defined by the NFV architecture introduce new failure points and mandate the need for additional testing methods to ensure the reliability of VNFs and services.

The following list summarises the new capabilities introduced by the use of virtualisation technologies and the proposed NFV architecture:

- Infrastructure sharing - VNFs run over shared compute, storage and network environment (contention);
- Workload dependency - Performance of the NFV Infrastructure is influenced by the type of load (network versus IT workload, CPU intensive, memory intensive or storage intensive) and number of VNFs executing;
- Use of I/O acceleration mechanisms - Special data plane acceleration mechanisms may be used for IO intensive applications;
- VNF Service chaining over multiple NFVI-PoPs - the path may consist of one or multiple VNFs, which may or may not be present on the same NFVI;
- Flexible resource allocation and scaling - A VNF will be instantiated with a defined amount of resources available to it. However, NFV allows for the MANO function to dynamically modify the amount of resources allocated to a VNF, as well as instantiate other VNFs, as the load requires;
- VNF mobility and failure recovery - VNFs may be transferred to another NFVI-PoP or instantiated to recover from failures.

3.1.3 Test Suite, Test Case and Test Profile

A test plan is a document which outlines the what, when, how, who, etc. of a testing project. It includes in general the objective and scope of the tests to be run. It serves as the general guideline for a testing project and for developing the test cases which are the tests themselves.

In 5GTANGO, the test plan is foreseen to be constituted from two parts of contribution: 1) from 5GTANGO which defines the generic scope and test objectives to test the compliance of VNF/NS regarding to ETSI-NFV specifications; 2) from the infrastructure provider who defines the test objectives specific to the NFV infrastructure on which the VNF/NS under test is to be deployed.

The test plan is then comprised of test cases which implements the test objectives defined in the test plan. A test case is a document which specifies the required test data, the preconditions, test steps, expected results and postconditions. A test case aims in general to validate one test objective, for example, the HTTP server returns a success code upon a correct message. As explained in previous section, test cases can be automatically generated from test models using MBT methodology and tool, which can save tremendously the effort regarding to manual test generation.

A test suite is a collection of test cases to test one specification that the SUT needs to be compliant with. For example, we can have a test suite which verifies the authentication feature of the SUT which can be decomposed to several single test cases to constitute the entire authentication
Figure 3.1: Test plan, test profile, test suite and test case

workflow. One test suite usually corresponds to a subset of test cases targeting the features or specifications of one aspect, such as functionality, performance, interface, security.

The test profile is defined as a subset of tests from the complete set of mandatory and optional tests that verify all the requirement of a specific type of product. Let’s take an example of a VNF which implements the firewall functionality. We can thus first define a product profile which is “firewall”. As a VNF, it need to be compliant with all the specifications for a generic VNF, and as a firewall, it needs to satisfy the functional and non-functional requirements towards a firewall software component. The product profile “firewall” needs to be verified and validated by a test profile “firewall” that comprises a set of tests that check all the requirements and specifications as a VNF and as a firewall software.

In 5GTANGO, test profiles are defined by the project regarding to the most frequent developed and deployed VNF or NS, such as vSwitch, firewall, or defined by the third party V&V platform provider.

Fig. 3.1 shows a foreseen example of the relationship between test plan, test profile, test suite and test case. Test profiles, test suites and test cases are developed according to the test plan(s) (represented by solid arrows in the figure). The test profile “vSwitch” includes two test suites which have different test objectives, and the test profile “Firewall” includes also two test suites. The test suite “Ve-Vnfm” belongs to both profiles. Test cases are chosen to be included in each of the test suite in order to accomplish the the objective of the test suite.

3.2 Test Result Management

The results of tests executed on the V&V testing platform need to be managed to facilitate the further use and analytics of the data. In this section, we will answer the following questions: what are the information that need to be included in test results, how they are presented and how they are stored for further use.
3.2.1 What Need to be Included in the Test Result?

From the 5GTANGO general requirements and the requirements of the pilots, the following information and data are required to be included in the test results:

1. Information about the test plan and test profile(s) applied on the target SUT. The test plan and test profile(s) provide information on what are the needed tests to be executed on the SUT, and in which order/scenario these tests should be executed. This information is essential because in industrial-grade applications (i.e. the smart manufacturing pilot in the project), it needs to be sure that different VNF implementation of the same feature (i.e. the software switch) have followed the same testing approach to make the test result directly comparable, which facilitates the NS developer to choose the most appropriate VNF for his/her NS;

2. The test traceability. The test traceability provides the matrix between the requirements and the tests that cover the requirements. It provides a clear visibility about the requirements that are verified by a VNF for VNF and NS developers, as well as for infrastructure providers;

3. The test environment configuration. It includes essentially the following data:

   1. Whether it is an emulated testing environment or a real one;
   2. The hardware information of the current test environment;
   3. The software resources and facilities made available for the test environment;
   4. Whether the SUT is isolated from other VNF or need to collaborate with other VNFs to complete the test;
   5. What are the test data provided and how they are sent to the SUT. For example, in the smart manufacturing pilot, we can distinguish two tests with one sending a high payload at lower frequency and the other sending low payload with high frequency.

4. The test result or conclusion. All the test results are linked with the test itself which is linked back to the test description included in the test plan and test profile. We can distinguish two types of test results:

   1. The functional test result. It consists of whether the test has passed/failed for the given test objective;
   2. The KPIs. It consists of non-functional test result such as the processing time of the given amount of data, the latency, etc. The results are either direct monitoring data provided by the monitoring module, or some data analytics result based on the monitoring data.

As mentioned early in the subsection, and it is worth noting again that the provided test result should be as directly comparable as possible in order to better support the decision-making process, either for the service platform provider, or for the NS developer.

3.2.2 Presentation of the Test Result: a Star System

The objective of V&V platform is to provide a set of comprehensive data on the tests executed on the SUT rather than to provide a conclusion on whether the SUT has good/poor performance or whether the SUT can be deployed on the target infrastructure. All the test data will be presented as they are in a machine-readable format (i.e. XML, JSON, YAML) for further data processing by other 5GTANGO modules.
However, the amount of raw test data is so vast that for a human being (for example, the developer), it is nice to have a piece of high level information that can be quickly caught and can abstract with more or less approximation the quality of the SUT after all the tests executed.

For this purpose, a **star system** is proposed to abstract the V&V raw test results. The principle of such abstraction is to score the SUT according to multiple test results and to map the score in a pre-defined classification in order to offer a general qualification of the SUT. For example, an SUT that successfully passed 90% of all the functional tests will be noted “5 stars”, whereas an SUT that passed only 50% of all the functional tests will get only “2 stars”.

To put forward such a star system, two specifications are essential:

1. **The scoring rules** specify which test contributes how many points in the total score;
2. **The classification** specifies the quality rating of the SUT regarding to the obtained score.

The final numbers of star will not try to substitute the raw test data to be submitted to the catalogue for metadata annotation. It’s only an approximative abstract that tries to give a general appreciation of the SUT.

In 5GTANGO, the specifications can be provided by the chosen **test profile**. For example, the V&V platform chooses for the SUT a test profile “software switch” that executes several test suites in a pre-defined order. There is a document of scoring rules associated with this test profile which calculates the score at the end of the test execution process. A classification document is also associated to the profile that make all the “software switch” VNFs that are tested under the same V&V platform comparable by reading the star number.

### 3.2.3 Test Results Repository

Test results are stored in a dedicated repository that enables that other modules or end-users (i.e. developer) to manage the test results.

The following are requirements towards the test repository:

1. The repository should provide a set of APIs that allows storing, retrieving, deleting, searching and updating test results;
2. All the test results and the star result of a SUT need to be retrievable as a package;
3. The test results need to be protected by applying security and privacy rules.

For the development of the test result repository, we believe the best approach is to use a NoSQL database due to the nature of the test result data format and structure. The final choice of technology will be taken once more detailed information of test result data will be available.

There is no assumption on the number of test result repositories that can be attached to one single V&V platform made for the moment. One repository is sufficient in most of the cases because the test results are retrievable and searchable regarding to the SUT, to the type of test (i.e. functional or performance), to the test execution time, etc. However, a V&V provider is free to make their own decision on the number of test result repositories.

Test results are sensible information and can create security and privacy issues. If some entity changes without any authorization the test results regarding to an NS to be deployed, the NS could damage the service platform during deployment or runtime if the NS is actually not conform regarding to the essential criteria of the service platform whereas the fake test results show the opposite. If someone else can see the test results of an SUT that the SUT developer does not intend to share,
it can damage the privacy of the developer. To protect the test result repository, the 5GTANGO general security framework which provide authentication and authorization functionality can also serve.
4 Monitoring in V&V

5G Networks are pushing new, more dynamic and flexible deployment scenarios that require the full automation of management and orchestration processes in heterogeneous, cloud-based, softwarized networking infrastructures. This paradigm requires the establishment of standardized validation and verification processes that will allow for the cross-platform (re-)utilization of (individual) VNFs, or an ordered set of them chained together to define a NS that would be able to be deployed in different VIM/NFVI environments and managed by different MANO frameworks. Part of this standardized validation and verification process is the definition of the tests with respect to the monitoring metrics and rules, the way that data is collected, as well as the processing and the delivery of the data to the respective V&V platform components that will take the decision (PASS/FAIL) with respect to the test execution.

In this respect, 5GTANGO V&V monitoring components will rely on the building blocks of SONATA Monitoring Framework [29], extending implemented functionalities in order to address these new requirements posed by V&V platform components and respective functionality.

In particular, the 5GTANGO Monitoring Framework will cater for:

1. The verification and validation related test execution on sandbox environment (supporting profiling);
2. The performance measurement of a NS deployed in a SP production environment (supporting policy and SLA management).

4.1 Interplay of Monitoring Framework with V&V Platform Components

As already stated above, for the purposes of 5GTANGO, the System Under Test (SUT) can be defined as either an individual VNF or several chained VNFs, defining a whole NS. Thus, monitoring data must be collected either from individual VNFs or from the NS, as part of the deployment on testing SP infrastructure for the purposes of V&V testing procedure. However, verifying functionality of a single VNF requires different criteria and includes different processes than verifying functionality at service level. Depending on the functions provided by a VNF or a NS, different test scenarios must be executed, based on respective standardized approaches and test methodologies. In this way, 5GTANGO V&V component will become a cross-platform component for MANO and NFVI implementations.

For the reader to position correctly the monitoring framework components as part of the V&V functionality cycle, we explain in details the steps the monitoring is involved in:

A VNF or a NS package is onboarded on the Catalogue of the sandbox environment or Service Platform, including VNF and NS descriptors with the set of metrics and policy expressions (defined in the form of value pairs of \{rule:threshold\} ). In the case of onboarded a VNF/NS on a Service Platform Catalogue for deployment on a production environment, the metrics and the policy expressions will take into account the results of the profiling and other testing execution metadata.

In some moment in time, the VNF or NS is deployed, either on a sandbox environment or on a production environment of a Service Platform, along with the set of monitoring tools that are
required for the measurement and collection of the metrics, as defined in the descriptors. The tools might be simple software libraries, programs, executables or scripts that may even come as part of the operating system distributions, e.g. ping or traceroute as standard libraries of Unix/Linux distributions, without the necessity to be installed during NS deployment. It is highlighted that SONATA already provides tools for metrics collection from four different types of sources:

1. Container exporter which runs inside the container of the VNFs to collect data from its performance;
2. VM exporter that collects data from Virtual Machines (VMs) supporting VNFs;
3. OpenFlow exporter which is a Python software that utilizes OpenDayLight API to collect data from the OpenFlow controller;
4. OpenStack exporter that has also been developed as a software module (in Python language) that uses OpenStack API to collect data from all OpenStack components.

Additionally, the plethora of VNFs and their chaining to provide a specific NS to be supported by the 5GTANGO ecosystem, dictates for the provisioning of tools that will allow the collection of additional metrics (even developer-specific ones) that span across OSI layers 2-5. In this respect, specific tools might be required for some VNF/NS testing and performance monitoring, e.g. iperf or other developer-specific scripts. In such cases, monitoring tools (also referring as monitoring probes) will be installed in an automated fashion. Depending on the tests to be executed, there might be the need either for the deployment of one auxiliary component (e.g. traffic generator) and measurements to be performed on the SUT, or the deployment of two auxiliary components where the second will collect data passing through the SUT. These auxiliary components can be considered as lightweight VMs or containers that do not belong to the NS but must be deployed in the sandbox environment only for V&V testing purposes.

Thus, the existing list of SONATA Monitoring Framework supported metrics will be enhanced in 5GTANGO, also including metrics in the form of active monitoring measurements, following, to the maximum extent, existing propositions from standardization bodies, especially those related to ETSI NFV WG, such as [12], [13] and [31], [8], [18].

Apart from static testing conditions, the monitoring framework must also provide the ability to the developer during testing and service provider during NS operation, to reconfigure the monitoring/testing parameters in order either to execute test(s) under different conditions (accommodating profiling operations) or to include new rules and/or modify thresholds on already existing rules (promoting Policy and SLA management operations). As part of reconfiguration, one can consider the ability to change the frequency of the measurements, apply new rules, increase the number of queries to a database, the HTTP requests on a webserver, traffic data on the network during run-time, etc.

After the data is collected in the time-series database of the monitoring framework, there is a need for a first-level of processing, especially to fulfill the requirements of Policy and SLA management. This capability is already part of the monitoring framework functionality realized through the functionality of an alert manager that collects the alerts in real-time and informs the interested (and authenticated/authorized) users for violations.

Finally, SONATA Monitoring Framework provides different ways for an entity to retrieve monitoring data, either as part of the testing execution or as part of the process of monitoring the performance of a NS on production environment. These ways of interaction with the different stakeholders, include websocket creation for real-time streaming of monitoring data, HTTP requests via a rich set of API commands for batch, asynchronous operations and the publishing of messages to a message bus in order for the subscribed entities to be informed.
4.2 A Representative Example of Network Service Test Execution on V&V Platform

With the emergence of the cloud paradigm and widespread adoption of server virtualization in data centers, most of these hosts are virtualized. Each hypervisor on a virtualized host adds another network layer in the form of a virtual switch. Realizing the importance of network-wide monitoring of flows, hypervisor vendors are increasingly adding support for network monitoring within the hypervisor virtual switch. Fig. 4.1 presents a basic diagram of the monitoring components included in NS testing process that is explained below.

In the general case, a Network Service might consist of two VNFs deployed in two remote PoPs, as depicted in the figure. VNFs can also be deployed by utilizing several recent technologies, including Virtual Machines and/or Linux or Docker Containers. Each software block has two network interface cards (NIC), one connected to a public network being used as a management network for remotely configuring virtual switches via the vSwitch Controller, based on testing conditions, defined in the Test Monitoring Controller, the second NIC being used to connect the two VNFs comprising the network service or one VNF with an auxiliary function/service (software module) deployed solely for the purposes of testing (e.g., when a web server reachability is tested via ping testing). The NICs are connected through proper LAN/WAN network, as shown in the figure.

Moreover, network monitoring protocols already existing in the routers can be supported. For example, OVS (Open vSwitch) already supports NetFlow and SFlow. Also, tools for the testing execution and monitoring metrics collection are automatically installed or come as part of the operating systems under test (such as iperf, ping, etc).
5 Test Execution Platforms

The V&V needs ways to execute the tests for functions and services that shall be verified by it. To do so, it utilizes one or more test platforms to which it connects. These test platforms are execution platforms that are capable to execute the network functions and services as well as the test cases attached to them, e.g., by sending test traffic to a service under test (SUT). Typical examples for such platforms are the 5GTANGO service platform, Open Source MANO, or the 5GTANGO emulator. However, the V&V platform is not limited to these platform implementations but offers an extensible abstraction layer (shim) that allows to interface with any platform as long as the needed interface wrappers are provided. Another important point is that 5GTANGO test platforms shall support testing of complete network services, composed of multiple VNFs, which allows to also identify problems that may be caused by interdependencies of different VNFs of a service [21]. In particular, a test platform can support one, some, or all of the following test classes:

- Functional tests;
- Reliability/stress tests;
- Integration tests (e.g. integration into different MANO stacks);
- Performance tests and benchmarks.

5.1 Deployment Flavors

A variety of deployment flavors for test platforms used by the 5GTANGO V&V are possible as shown in Fig. 5.1. It is important to understand that a single V&V deployment can connect to multiple test platforms. Additionally, third party test platforms might offer multi-user support which means they can be used by different V&V instances at the same time.

5.1.1 Development Test Platform

This is the most simple test platform setup that can be used by the 5GTANGO V&V. It is installed locally on the developer’s laptop or on a small server and allows the network service developer to locally verify new services before they are on-boarded to an external service or test platform. A development test platform installation is in particular suited for functional tests which do not depend on the performance of the SUT. However, its usability for performance testing and benchmarking is limited since the test results are not representative for a deployment on real, production-ready infrastructure. Nevertheless, basic insights about the performance behavior of a service might still be derivable, e.g. does the performance of a particular VNF of a service improve if more cores are added [23].

Technically the development test platform might either be realized by using an NFV infrastructure emulation approach [22] [24] or a minimal cloud installation, e.g., single node OpenStack.
5.1.2 Private Test Platform

The private test platform deployment flavor implements a full-featured test execution platform that lives in the same administrative domain as the V&V itself. Such a platform can, for example, be installed in a mobile operator’s premises and is exclusively used by this operator. The purpose of such a platform is two fold. First, the platform can be used to validate the operator’s own services. Second, it can be used to validate third party services that shall be deployed on the operator’s infrastructure. This deployment option provides several benefits. First, the operator can fully trust the platform and thus the test results that are obtained by it. Second, the platform can be designed to be a one-to-one replication of the operator’s production platform. Using this, performance tests and benchmarking results directly indicate the resulting performance of a service when it is deployed on the production platform.

Technically the private test platform is a full-featured NFVI solution that can either be deployed on dedicated hardware or share the hardware with the operator’s production platform, e.g., by using a cloud-in-cloud deployment approach [20].

5.1.3 3rd Party Test Platform

Another deployment flavor for test platforms is called 3rd party test platform and describes the deployment of a test platform by a third party which can then be accessed by remote V&V platforms. This deployment option offers new business models, e.g., test-platform-as-a-service (TPaaS), in which test facilities are offered to customers and can be used on-demand and be scaled as needed, e.g., by offering different pricing models for a single test run or multiple concurrent test run as known from continuous integration (CI) provides, like Travis-CI [11].

There are two major options for such a third party deployment as shown in fig. 5.1. First, the third party deploys a V&V component as well as the connected test platform and directly offers verification and validation services to its customers. The other option is to only deploy the bare test platform and let remote V&V components connect to it. The main benefits of this option are the reduced setup and maintenance efforts for the customers of the test platform. The shortcomings of
this deployment flavors are, on the one hand, the limited comparability of, e.g., performance test results if the test platform operator uses entirely different hardware than its customers. On the other hand, is a trust model between V&V and test platform as well as its customers necessary to ensure the integrity of test results.

Technically, a 3rd party test platform typically implements a full-featured NFVI platform that might offer different types of interfaces to its remote users. In general, a test platform operator should provide information about the used infrastructure to ease the comparability of performance test and benchmarking results.

5.2 Annotating Test Results with Platform Information

There can be different test execution platforms which might be composed of, for example, different hardware, other software components, or just use different configurations which turns the comparison of test results into a complicated task. This is especially true for performance tests which heavily depend on the underlying execution platform. To tackle this problem, additional test execution platform information is always stored together with the actual test results (see sec. 3.2). These additional information can for example contain if the test platform was a emulated environment or a real platform, which MANO solution was used to orchestrate the SUT, or on which hardware boxes the platform is running. Such information has either to be provided manually, once the test platform is added to a V&V or automatically requested from the test platform as such through an appropriate interface. This will help developers as well as service operators to understand and interpret test results. It will even be possible to predefine standard setups for test platform which can be considered when a new test platform is created. This will also allow to certify test platforms that follow the standards.
6 Catalogue and Metadata Management

This section focuses on several adjacent 5GTANGO components of V&V. They either produce metadata for 5GTANGO and the V&V, uses the V&V to produce more metadata, or manage the data output by V&V.

6.1 5GTANGO Catalogue

6.1.1 Goal of the 5GTANGO Catalogue

The 5GTANGO Catalogue is considered to be a multi-sided catalogue, addressing different stakeholder needs. It supports developers by providing a repository for storing the developed VNFs/NS, while also annotating them with additional information (as metadata) which can be used to enhance its key functions and interfaces for storing, searching and retrieving VNFs/NS based on these metadata, as well as providing added value services like Decision Support and Continuous Optimization.

The V&V results, the profiling outcomes, information related to policies and QoS (through SLAs) will be stored as metadata of the corresponding VNFs that will be utilized for all its operations. It should be noted, that the metadata will be updated with new V&V results as well as with information from runtime monitoring.

Moreover, the decision support and continuous optimization services will analyse and correlate the aforementioned information providing their outcomes to several components. The decision support mechanism supports the operators by providing information related to the optimum selection of VNFs/NS based on their requirements, while the continuous optimization mechanism will interact with the V&V framework in order to trigger new V&V tests (feedback loop towards V&V) by considering the actual outcomes and behavior of the deployed VNFs/NS (through the actual deployment information and the collected monitoring data).

6.1.2 Metadata

The functionality of the 5GTANGO Catalogue is heavily based on metadata for all stored objects, allowing efficient search, correlation between different stored objects, versioning, updates etc. Metadata are also needed for the Catalogue’s decision support and continuous optimisation mechanisms.

The following key categories of metadata, that the catalogue will need to support, have been identified:

- **Description of the NS functionality**
  This includes any information that will describe how the NS functions (e.g Included VNF(s), deployment pattern, developer information, version, description, etc.).

- **Infrastructure Description**
  Includes information about the infrastructure the NS was tested and deployed (e.g. location, constraints/capabilities).
• **Description of Policies**

The metadata that describes the policies includes information about resource constraints (e.g. minimum CPU, memory, disk), needs for special characteristics (e.g. encryption, high mobility), scalability aspects (support for horizontal scaling, need for vertical scaling on a set of rules), VNF specific metrics or VL (Virtual Link) specific metrics (e.g. end-to-end delay between VNF1 and VNF2, average response time for an http server).

• **V&V Test Results**

The V&V framework will generate a set of results about executing the VNFs/NS on specific infrastructure information (e.g. test configuration information like what are the tests performed, what are the available resources on the test execution platform etc.).

• **Production Results**

Any information about the results of executing the VNF/NS on real world infrastructure. This is monitoring data and includes VNF/NS performance metrics (e.g. CPU, memory, disk, network usage), Infrastructure performance metrics, VNF/NS status information, Infrastructure status information.

• **SLA description**

All information regarding the SLA components is included in this category including the base SLA information (e.g. SLA sign date, duration, stakeholders, covered NS/VNFs descriptions, roles and responsibilities), metrics (e.g. availability, with definition and usage of them and their underlying measurements).

• **VNF Base Information**

This includes, name of VNF, version, SDK version used, author information (name, company, classification e.g Certification of trusted developers or certification of v&v providers), licensing, dependencies, target group (e.g. small scale large scale), number of issues and maintenance related information. This maintenance information may include number of releases, when the VNFs/NS were first released, when were they last updated/modified (changelog) etc.

• **Profiling information**

Profiling information regards profiling of an individual VNF as well as profiling of an overall Network Service (NS). Profiling information may be collected via black-box or white-box testing. Such information can be provided upon the realisation of a set of tests and updated accordingly upon collection of further monitoring data based on NSs execution.

In both cases, profiling data is going to consider information regarding resources usage, Quality of Service (QoS) and Quality of Experience (QoE) characteristics, VNF specific metrics (e.g. http\_avg\_response\_time, number of packets processed per second) and NS specific metrics (e.g. end-to-end delay, jitter, aggregate bandwidth consumption), qualitative characteristics (e.g. stability, fault-tolerance, reliability) as well as characterization of a VNF or NS in terms of Compute/Network Burstiness, or Compute/Network/Storage Intensiveness.

In more detail, a set of indicative metrics include:
VNF profiling metrics: avg/min/max CPU usage, avg/min/max memory usage, storage requirements, inbound/outbound traffic served per interface per number of users, deployment constraints (e.g. min CPU, min memory, collocation constraints);

NS profiling metrics: aggregated bandwidth consumption, end-to-end delay, jitter, link-based characteristics (e.g. exchanged traffic), VNF dependencies-based characteristics (e.g. software/API calls among software components);

Qualitative metrics in VNF and NS level: number of faults, misbehaviour of a VNF, average time for recovery from faults, mitigation effects of a failure, virtual links stability;

Adoption/Business metrics: number of deployments of a VNF or NS, number of associated policies and SLAs, QoE perceived by end users.

The collected profiling information is going to be made available in the form of a descriptor (profiling descriptor) of the VNF or the NS. The profiling descriptor may be taken into account by a software developer or a service platform manager towards the definition or refinement of deployment or runtime policies.

6.1.2.1 Lifecycle of Metadata

The metadata to be stored and analysed (as described above) include metadata both for VNFs and for NS. These are briefly summarized as follows:

**VNF**

- VNF base information;
- TTCN3 test cases;
- Monitoring information.

**NS**

- Metadata of all the included VNFs and VLs and VNFFGs;
- Profiling;
- “End-to-end” parameters;
- Policy;
- SLAs;
- Monitoring information.

Regarding the lifecycle of the metadata, it is initiated by the developer that provides the VNF base information and the test cases in TTCN-3 format through the SDK. The developer composes VNF(s) in a NS, which contains the metadata from all the VNF(s) as well as metadata about the virtual links (VLs) and the VNF forwarding graphs (VNFFGs). The NS gets profiled by the developer to generate profiling results. These results are considered by the developer in order to provide the overall / “end-to-end” parameters of the NS (captured in the corresponding descriptor). This information is also exploited for the creation of policies by the developer for the specific NS. The service provider obtains those policies, alters them accordingly, and proceeds with the SLAs for the NS. Post deployment, the NS and the VNF(s) that comprise it are monitored by the service.
provider. Based on the monitoring information, the envisioned feedback loop may be triggered to re-execute the same or different tests. The latter affects the overall metadata lifecycle since new test cases will be triggered, resulting to new profiling information, new descriptors, new profiles and policies. This lifecycle is illustrated in fig. 6.1.

6.1.3 Added Value Mechanisms

The 5GTANGO catalogue goes beyond a simple data store by offering two added value mechanisms that exploit the metadata in order to automate the V&V lifecycle (Continuous optimisation) and help the operators choose the most suitable VNFs/NS for their needs (Decision support).

6.1.3.1 Decision Support

Nowadays, the number of services to choose from is increasingly overwhelming. Therefore, there is a need for filtering, prioritizing and efficiently deliver information in such way that supports the decision making process of the users. The process of optimum selection of VNFs/NS by the operators is no exception and there is a clear need to facilitate their decision making process by providing suggestions that match their preferences and needs. To solve this problem, the Catalogue
Figure 6.3: Continuous Optimisation

offers a decision support mechanism, which aims at supporting the infrastructure owner / operator regarding the selection and use of assets (i.e. VNFs and NS).

Given the fact that the Catalogue will not only store VNFs/NS but also annotate them with metadata that are coming from the different 5GTANGO components (e.g. SLAs coming from SLA management framework), it can analyze and use this information to propose specific VNFs/NS to the operators based on their requirements.

Illustrated in fig. 6.2, by analyzing the SLA and Policy metadata, the mechanism identifies the QoS/QoE requirements of the operators generating useful statistics by aggregating the data and making a QoS history profile for every operator. The QoS history of operators can be utilized to effectively implement recommender system techniques like content based filtering or item based collaborative filtering based on QoS predictions and matching them to corresponding assets by searching the stored metadata.

These metadata also include information like VNF/NS version, developer and licensing information, number of issues, maintenance related information like number of releases, initial release date, last update date (SDK), test cases and test outcomes (V&V component), deployment configurations (service platform), monitoring data, QoE data (infrastructure components), policies (policy management framework) and SLAs (SLA management framework) annotated to the corresponding VNFs/NS.

6.1.3.2 Continuous Optimisation

The purpose of the continuous optimization service is to introduce dynamicity in the overall V&V process by allowing not only static (i.e. pre-defined) test configurations to be executed, but also by considering the actual outcomes and behavior of the deployed VNFs/NS and proposing new V&V tests to be executed accordingly. In order to achieve this, the continuous optimisation mechanism obtains as input the monitoring information both from the test / qualification infrastructure and from the production infrastructure. Based on differences observed between the data generated by real-world and the test scenarios and by also considering the previous test configuration parameters the mechanism will propose new V&V tests to be triggered, going back to the V&V framework and thus causing a “feedback loop”. In addition to triggering tests, the data will be sent back to the developer to help them better understand how their VNFs/NS perform (shown in fig. 6.3).

The major challenges to overcome when building this mechanism will be how the large amount of monitoring data will be processed and how the tests will be generated based on the analytics obtained. The goal when processing the monitoring data is to find actionable differences between the two sets of data that we have for the various metrics that are stored in the catalogue. The test generation will use the information about those differences as well as information related to the test scenarios that were run (including information about the test parameters that were used) in
order to propose new V&V tests that will be “closer to reality”. This information will also be sent back to the SDK and it will be presented to the developers as a “performance report” for their VNFs/NS.

6.2 Policies Management Framework

In 5GTANGO, a policies management framework is designed aiming to support the set of orchestration mechanisms provided by the 5GTANGO service platform. The policies management framework interplays with the SLAs management framework. High level business-oriented policies are described in the form of SLAs and are translated to set of lower level technical-oriented policies, tackled by the policies management framework.

Policies may regard real-time management of network services provision, tackling aspects related to scalability characteristics (horizontal and vertical scalability) of the deployed VNFs, management of the allocated infrastructural resources, adaptation of VNF specific metrics aiming at optimal provision of the service, triggering of alerts as well as activation of VNFs upon specific conditions in the network. In addition to real-time management objective, policies may be used for dynamic management of the created network slices (e.g. satisfaction of isolation requirements, dynamic management of allocated resources in the slice) as well as provision of guidance/recommendation to mechanisms applied for specific cases (e.g. tackling of aspects related to edge computing scenarios, supporting workload orchestration).

Software developers, infrastructure owners, slice managers and service platform managers may interact with the policies management framework, as follows:

- Software developer: introduces set of metrics and policy expressions as metadata in the developed software (e.g. virtual network function, network service). The set of metrics and expressions can be adopted towards a policy formulation by the service platform operator;

- Infrastructure owner: introduces set of constraints or business models towards the reservation of compute, storage and network resources that may be taken into account towards policy formulation by the service platform operator;

- Slice manager: provides specifications for the creation, maintenance and dynamic management of a network slice. Such specifications may be used by a policy for network services placement and orchestration aspects;

- Service platform manager: formulates a policy taking into account input provided through the software developer, the infrastructure owner and the slice managers. Associated a policy or a set of policies with a network service.

The following workflow is considered:

- The software developer provides set of metadata in the software package that is associated with a NS, as it is provided by the SDK. The software package includes descriptors for the NS, the set of VNFs, VLs and VNF-FGs. For each case, generic as well as custom metrics are defined along with expressions that can be included in a policy. Each expression consists of a “rule - condition - action” triple;

- The software package of the NS is made available at the V&V. Testing, profiling, validation and verification is realised. The software package is certified and ready for adoption by service providers;
The service platform manager decides to formulate a policy and associate it with a network service. Through a policies editor, he is able to get the set of provided metrics and expressions already included in the software package by the developer, adopt, modify or even extend them. Further expressions can be included in a policy tackling network slice management or infrastructure management aspects. The service platform manager may also consider set of constraints or reservation schemes provided by infrastructure owners. According to the adopted reservation scheme, set of expressions may be introduced for guaranteeing the appropriate usage of resources;

During the placement of a network service, the selected policy is activated. This action leads to the activation of a set of monitoring probes providing real time data for the required metrics. In addition to the collection of monitoring data, it is examined whether the conditions in the declared rules (e.g. whether a metric is below/above a specific threshold) are satisfied or not, leading to the triggering of specific alerts. The set of produced alerts -along with any further monitoring data- are provided to the policy inference engine for reasoning purposes. Such alerts and data are actually send to a message broker where they are consumed by the policy inference engine. The inference engine supports reasoning over the provided data based on the declared rules in the policy editor. Real time inference along with conflict resolution functionalities based on the definition of priorities in the declared rules are supported. The inference engine instantiation is going to be realised through a plugin that is going to consist part of the service platform;

According to the produced inference results, set of actions are suggested to be taken. Such suggestions are taken into account by the relevant orchestration mechanisms (realised through

Policies description is going to be based on the definition of a formal language. The main objective is the specification of a language that is going to facilitate definition of expressions that can include actions binded to the set of orchestration mechanisms that can be provided by NFV orchestrators.

### 6.3 SLA Management Framework

Software Networks include high performance Virtual Network Functions (VNF) and Network Services (NS) that require a plethora of system resources. These technologies are essential to support many aspects of the anticipated functionality offered by the 5G networks, and probably will be open in a distributed environment. The current vision and research challenges in 5G go beyond the focus on the underlying infrastructure, towards the stakeholders of the ecosystem, including SLA Management. The management of virtualized networks focuses on maintaining specific quality levels. Therefore, Quality of Service (QoS) expectations are driving end-users to negotiate specific QoS levels with their service providers through Service Level Agreements (SLAs).

An SLA is defined as a contract between the service provider and the end-user that specifies the QoS level that can be expected. Considering that the quality requirements submitted by the end-user are stated into the SLA, it becomes essential for the service providers prior to signing the specific SLA, to estimate the resources needed to fulfill the user requirements for every application that is offered as a service.

The SLA Management Framework plays a significant role, in mapping the end-user’s defined service high-level requirements (workload parameters), and policies expressions (low-level requirements) to the resource level attributes.

A vital part of the SLA Management Framework is the SLA Generator that firstly creates the SLA templates for the Service/Infrastructure Provider, and secondly creates the final SLA itself. This Generator will be able to obtain:

- A set of associated policies for a specific NS;
- V&V results;
- Appropriate descriptors from both the profiler and the SDK, all in metadata format accessible from either the 5GTANGO Catalogue or the Message Bus, analyze them and create an SLA template in an automated way.

For the NS to be deployed in the service provider’s infrastructure, several steps must be implemented. The roles that are foreseen in this process include the service provider, the infrastructure provider and the end-user. Apart from these roles, an important component for the SLA Management Framework is the Policies Framework. The Mapping Mechanism that is included into the SLA Management Framework, will translate the high-level parameters to low-level, while at the same time decompose service level objectives to associated policies for better Policy Enforcement.

The input dataset is determined through information obtained as metadata - through the Message Bus or the Gatekeeper. All the information is provided by the Policy Framework, the Service/Infrastructure Provider and the End-User. In more detail, each one of them is responsible for the following inputs:

- Policy Framework defines a set of policy rules associated to the VNF/NS;
- Service/Infrastructure Provider defines a set of low-level requirements constraints as metadata;
End-User introduces a set of high-level requirements – VNF/NS workload parameters also as metadata. The mapping between the high-level requirements and the low-level takes place. To this end, the result of the mapping output is the QoS parameters that are needed to be expressed in the SLAs and needed for the Policy Enforcement. This information is available to the SLA Generator, being the basis of the final SLA.

In addition, an enhanced input should be defined through the Monitoring Framework, with metrics from V&V outcomes on sandbox environment and also the performance measurement of a NS deployed in a SP production environment.

It should be noted that the mapping results are formatted as metadata in order to be stored into a mapping repository. The goal is the mapping results:

- To be available for the SLA framework for a future selection of the appropriate SLA terms without unnecessary processing;
- To be accessible from the 5GTANGO Catalogue and manageable for its functionalities.
7 Conclusion

This deliverable presented the strategy of 5GTANGO V&V platform that will enable the DevOps approach for 5G environment. It is now conceptual and high-level and will be operationalized in the forthcoming months.

7.1 Scope and Architecture

The target of the 5GTANGO V&V platform is VNFs (Virtual Network Functions) and Network Services, which are graphs of one or more VNFs. They will be tested and certified by the V&V platform regarding to requirements defined by the 5GTANGO or by service operators. The V&V platform works closely with other 5GTANGO components such as the Catalogue, the SDK and the service platform which establish together the 5GTANGO DevOpes ecosystem. The proposed architecture is designed to be flexible and adaptable that aims to meet the requirements from different stakeholders. Functionalities are considered as plug-ins to implement a modular system, and APIs are considered to allow the V&V platform to plug the appropriate test execution platform as multiple test execution platforms are available offered by different stakeholders.

7.2 Methodology and Approach

As the name suggests, the V&V platform aims to provide a proof of validation and verification, not to provide debugging details for developers. Blackbox and whitebox testing approach will both be considered, but whitebox approach is only used in the Network Service Profiling which refers to testing and analyzing the performance of an NS knowing it’s internal graph. The alignment with ETSI NFV recommendations and specifications is also taken into consideration, which lead us to choose TTCN-3, which is a standardized test scripting language widely known in the telecommunication sector.

The tests needed for V&V can be generated automatically using the MBT (Model-Based Testing). Tests are categorized and stored in a repository to facilitate the composition of test suites to be executed and the prioritization of test execution if the test execution resource is limited. Test results must provide as complete as possible information on the system under test and aim to offer a direct comparability between different implementation of a same VNF. Monitoring functionality is also essential to collect the VNF/NS performance data. This module will extend the existing SONATA monitoring component to meet 5GTANGO’s requirements.

7.3 Catalogue and Metadata Management

To fully achieve a DevOps environment, the V&V platform is essential, but what to provide to the V&V as input and how to make use of the data output by V&V is also indispensable.

The 5GTANGO Catalogue is considered to be a multi-sided catalogue, addressing different stakeholder needs. It supports developers by providing a repository for storing the developed VNFs/NS, while also annotating them with additional information (as metadata) which can be used to enhance its key functions and interfaces for storing, searching and retrieving VNFs/NS based on
these metadata, as well as providing added value services like Decision Support and Continuous Optimization.

The 5GTANGO policies management framework is designed aiming to support the set of orchestration mechanisms provided by the 5GTANGO service platform. The policies management framework interplays with the SLAs management framework. High level business-oriented policies are described in the form of SLAs and are translated to set of lower level technical-oriented policies, tackled by the policies management framework. The SLA management framework plays a significant role, in mapping the end-user’s defined service high-level requirements (workload parameters), and policies expressions (low-level requirements) to the resource level attributes.

### 7.4 Future Works

This deliverable provides a general guideline for the 5GTANGO V&V framework, which will be implemented in several work packages:

- WP3: implementation of core verification and validation functions;
- WP3: implementation of the catalogue and metadata management;
- WP4: service model and tests implementation.

And more details of the 5GTANGO overall architecture will be described in WP2 that helps to clarify the position and functionality of the V&V platform from a different point of view.
8 Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Defined as</th>
</tr>
</thead>
<tbody>
<tr>
<td>3GPP</td>
<td>3rd Generation Partnership Project</td>
</tr>
<tr>
<td>5G</td>
<td>Fifth generation of mobile networks</td>
</tr>
<tr>
<td>5GPPP</td>
<td>5G Infrastructure Public Private Partnership</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>BSS</td>
<td>Business Support System</td>
</tr>
<tr>
<td>CI</td>
<td>Continuous Integration</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CRUD</td>
<td>Create Retrieve Update Delete</td>
</tr>
<tr>
<td>DevOps</td>
<td>Development and Operations</td>
</tr>
<tr>
<td>DUT</td>
<td>Device Under Test</td>
</tr>
<tr>
<td>EPC</td>
<td>Evolved Packet Core</td>
</tr>
<tr>
<td>ETSI</td>
<td>European Telecommunications Standards Institute</td>
</tr>
<tr>
<td>FSM</td>
<td>Function-Specific Manager</td>
</tr>
<tr>
<td>FG</td>
<td>Functional Group</td>
</tr>
<tr>
<td>IO</td>
<td>Input Output</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>ISG</td>
<td>Industry Specification Group</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet Service Provider</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>KPI</td>
<td>Key Performance Indicator</td>
</tr>
<tr>
<td>LTE</td>
<td>Long Term Evolution</td>
</tr>
<tr>
<td>MANO</td>
<td>NFV Management and Orchestration</td>
</tr>
<tr>
<td>MBT</td>
<td>Model-based Testing</td>
</tr>
<tr>
<td>NF</td>
<td>Network Functions</td>
</tr>
<tr>
<td>NFV</td>
<td>Network Function Virtualization</td>
</tr>
<tr>
<td>NFVI</td>
<td>NFV Infrastructure</td>
</tr>
<tr>
<td>NFVO</td>
<td>NFV orchestrator</td>
</tr>
<tr>
<td>NIC</td>
<td>Network Interface Card</td>
</tr>
<tr>
<td>NS</td>
<td>Network Service</td>
</tr>
<tr>
<td>NSM</td>
<td>Network Service Manager</td>
</tr>
<tr>
<td>OASIS</td>
<td>Advancement of Structured Information Standards</td>
</tr>
<tr>
<td>OCL</td>
<td>Object Constraint Language</td>
</tr>
<tr>
<td>OSS</td>
<td>Operations Support Systems</td>
</tr>
<tr>
<td>OVS</td>
<td>Open vSwitch</td>
</tr>
<tr>
<td>QoE</td>
<td>Quality of Experience</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RCA</td>
<td>Root Cause Analysis</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>RPC</td>
<td>Remote Procedure Call</td>
</tr>
<tr>
<td>SDK</td>
<td>Service Development Kit</td>
</tr>
<tr>
<td>SDN</td>
<td>Software Defined Networking</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>SP</td>
<td>Service Platform</td>
</tr>
<tr>
<td>SPoF</td>
<td>Single Points of Failure</td>
</tr>
<tr>
<td>SSM</td>
<td>Service-Specific Manager</td>
</tr>
<tr>
<td>SUT</td>
<td>System Under Test</td>
</tr>
<tr>
<td>TOSCA</td>
<td>Topology and Orchestration Specification for Cloud Applications</td>
</tr>
<tr>
<td>TTCN-3</td>
<td>Testing and Test Control Notation Version 3</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>V&amp;V</td>
<td>Verification and Validation</td>
</tr>
<tr>
<td>VL</td>
<td>Virtual Link</td>
</tr>
<tr>
<td>VIM</td>
<td>Virtual Infrastructure Manager</td>
</tr>
<tr>
<td>VLAN</td>
<td>Virtual Local Area Networks</td>
</tr>
<tr>
<td>VNF</td>
<td>Virtualized Network Functions</td>
</tr>
<tr>
<td>VNFD</td>
<td>VNF Descriptor</td>
</tr>
<tr>
<td>VNFFG</td>
<td>VNF Forwarding Graph</td>
</tr>
<tr>
<td>VNFM</td>
<td>VNF Manager</td>
</tr>
<tr>
<td>Acronym</td>
<td>Defined as</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>WIM</td>
<td>WAN Infrastructure Manager</td>
</tr>
<tr>
<td>WG</td>
<td>Working Group</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
<tr>
<td>YAML</td>
<td>YAML Ain’t Markup Language</td>
</tr>
</tbody>
</table>
A Bibliography


[19] NGMN. Ngnm 5g white paper. 2015.


[26] PragmaDev. pragmadev.


