Network Service Descriptors as a Factory

Agile Networking for Differentiating MSO Business Services

A Technical Paper prepared for SCTE/ISBE by

Hans Vanderstraeten
Team Lead Virtual Networks Orchestration
Nokia
Copernicuslaan 50, B2018 Antwerp
323-240-7905
hans.vanderstraeten@nokia.com

Erwin Six, Lead Solution Architect VNO, Nokia

Mihai Fagadar, R&D Team Lead VNO, Nokia

Willem Acke, Lead Solution Architect VNO, Nokia

Table of Contents

Title **Page Number**

List of Figures

Title **Page Number**

Introduction

Business Services have been a new ambition for MSOs over the past decade. The Service Level Agreements, among others, associated with connectivity services, imply typically customer delivery lead times of weeks, up to months when the MSO must go off-footprint. By contrast, the same business goes to the Cloud Service Provider, and gets compute and storage infrastructure at his disposal in minutes. These businesses are soon expecting network services to be delivered together with compute and storage, in minutes. SD-WAN delivers on these promises, and drives a need for end-to-end Network Orchestration.

Orchestration has become an overloaded word in the industry. While Network Orchestration and Application Orchestration share a lot of technologies coming from the domains of SDN and NFV, the two approaches serve different objectives. The paper will specifically address Network Orchestration, aiming at launching new revenue generating Business Connectivity services rapidly, with Value-Added Services running in private or public cloud data centers chained-in into the Network Service. MSOs will be able to tap into new revenue streams through the extension of their addressable market going off-footprint, adding new Value-Added Services to their product portfolio, and becoming the trusted middle man in the value chain between Enterprises and public cloud service providers.

The rapid market traction of SD-WAN overlay IP networking, enabled by product suites such as Nuage, fuels the requirements for Network Orchestration: overlay to underlay connectivity, connectivity into Public Cloud, Internet Break-out, flexible onboarding of virtualized Value-Added Services (VAS) such as firewalls, wan optimizers, email filters, and others, all easily customizable for each specific enterprise customer of the MSO.

Various Network Orchestrators, open source and others, have emerged. However, while these all enable writing a Network Service Descriptor (NSD) for a specific service chain use case, including a specific VAS over a purpose-built plug-in, significantly more is needed to deliver Network Services and NSDs rapidly, massively, and reliably. An approach is required that allows to produce NSDs as-a-factory, rather than just 'executing an NSD'.

The paper will discuss a combination of approaches that together allow to deliver Network Services by MSOs in the most competitive way, and more specifically, how higher automation can be achieved through a 'best-practice'-based abstraction of enterprise services. A combination of novel Communication and Software technologies are combined to, as examples, automatically generate TOSCA types and plug-ins out of (YAML-based) API specifications; write complex NSDs simply and reliably through substitution and decomposition; and manage a library of NSDs through release upgrades of plug-ins through an extended suite of Continuous Integration and Continuous Testing.

On Orchestration and Virtual Networks

Orchestration is a term frequently used when referring to Software Defined Networking (SDN) and Network Functions Virtualization (NFV) solutions. In essence, Orchestration aims to answer the problem of service agility, service velocity and OPEX control in both network connectivity and application domain. However, different meanings are associated with the word and different types of orchestration are emerging with different domains, roles and responsibilities.

1. ETSI MANO Orchestration

In the communications networks context, including SDN and NFV, orchestration refers to the automation of order fulfillment, service assurance, as well as other business processes in a software-controlled environment. Orchestration of previously manual processes is key to unlocking the OPEX savings expected from SDN and NFV, where moreover each of the SDN and NFV functionalities being orchestrated are autonomous systems on their own. The term alludes to a conductor who orchestrates music for an orchestra and ensures that all the musicians play in tune and rhythm. In turn, a Network Service Designer is the composer determining the notes to be played by each SDN controller, Virtualized Network Function Manager (VNFM), and any other Networking Function, while the Conductor or Orchestrator directs the overall performance of the orchestrated composition.

Further, SDN and NFV do promise for a much more easy consumption of applications and networks, as MSOs and other Communication Service Providers (CSPs) are being challenged by their customers (enterprises and residential, fixed and mobile) to be 'satisfied instantaneously' – the audience being the ultimate judge of the performance of the orchestra.

Standardization forums such as TM Forum, ETSI, Metro Ethernet Forum (MEF), the Open Networking Foundation (ONF), and others cover parts of the orchestration spectrum each with a different focus. Some are more specific about network connectivity, some are more specific about Virtual Network Functions (VNFs), and some are more specific about service decomposition.

ETSI proposes the following reference architecture in the NFV domain:

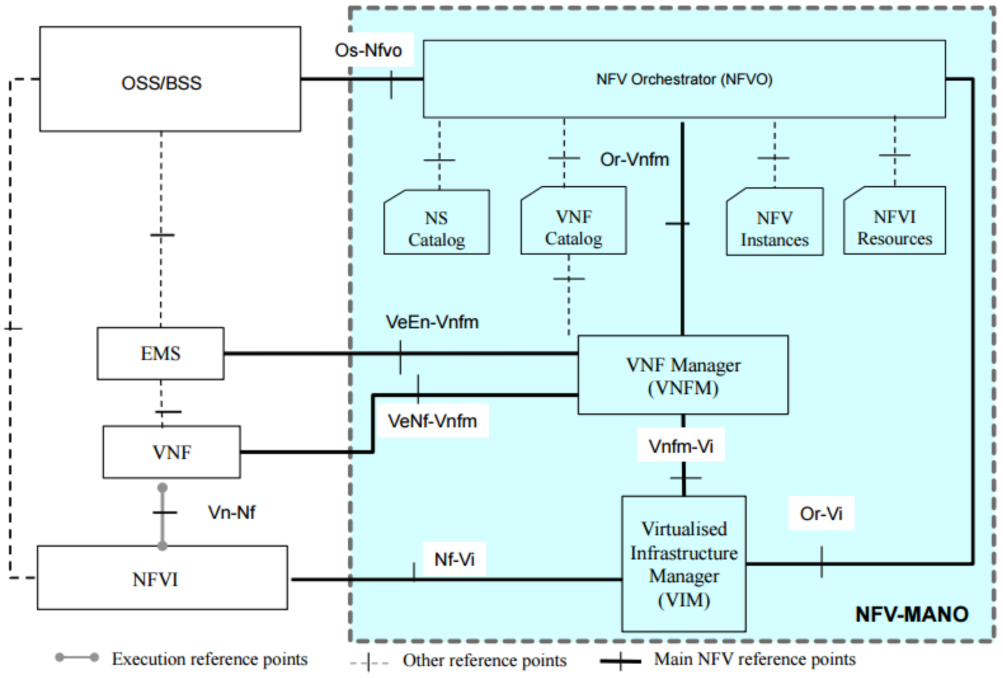

Figure 1 – ETSI MANO Reference Architecture

- NFV Orchestrator (NFVO): Responsible for on-boarding of new Network Services (NS) and Virtual Network Function (VNF) packages; NS lifecycle management; global resource management; validation and authorization of Network Functions Virtualization Infrastructure (NFVI) resource requests;
- VNF Manager (VNFM): Oversees lifecycle management of VNF instances; coordination and adaptation role for configuration and event reporting between NFVI and E/NMS;
- Virtualized Infrastructure Manager (VIM): Controls and manages the NFVI compute, storage, and network resources.

Note that in the ETSI MANO NFV framework, there are essentially two orchestrators: the VNFM as an Application Orchestrator, and the NFVO as a Network Orchestrator. Further, the ETSI framework applies to Virtualized Network Functions, and therefore spans the Datacenter domain only.

Extending upon the ETSI MANO definition, Figure 2 adds the Service Orchestration (SO) layer to the Business Service level which essentially concatenates several Network Services with Physical Network Functions or their management agents, bridging the legacy and virtual domains.

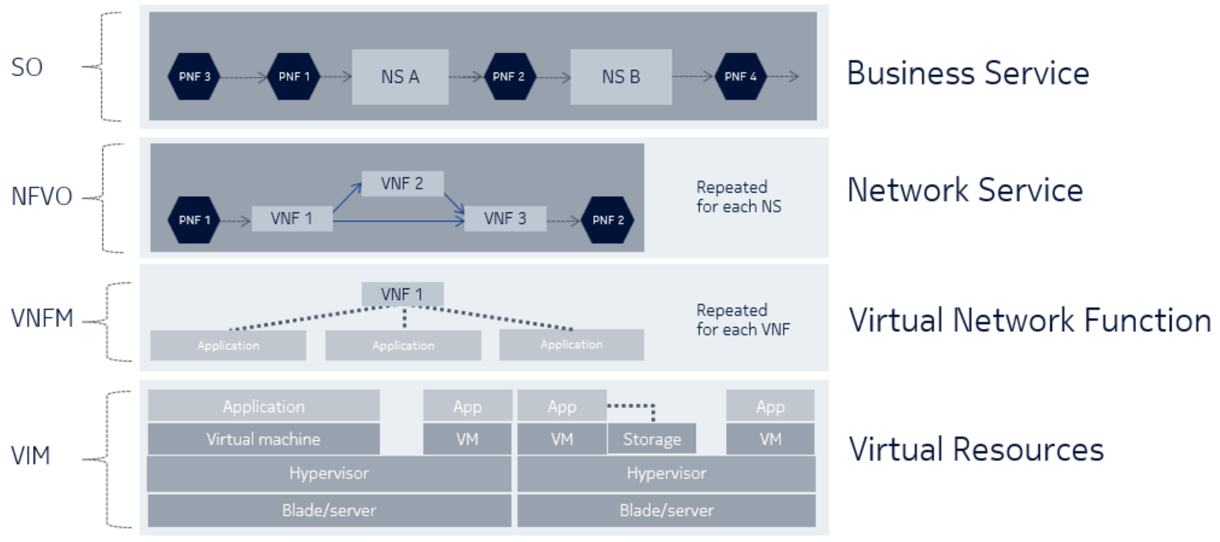

Figure 2 – Service delivery terminology (ETSI-MANO)

Given that orchestration can be defined at different layers and domains, a central concept to come to a meaningful split of responsibilities among the orchestrators is “separation of concern”. A Network Orchestration problem is divided across domains, where each of the domains is defined by the best practices and potentially organizational split of accountability and responsibility. Therefore, the Network Service Orchestration is essentially a well-structured collection of the domain-specific orchestrators, managers, and controllers. Each orchestrator decomposes and automates services, resources, and tasks in their respective domains.

2. Extending Orchestration

In a collaborative effort with the most of the industry actors and vendors, Verizon has brought forward a very specific proposal for the above mentioned "well-structured collection". Figure 3 is an extract of this public Verizon document:

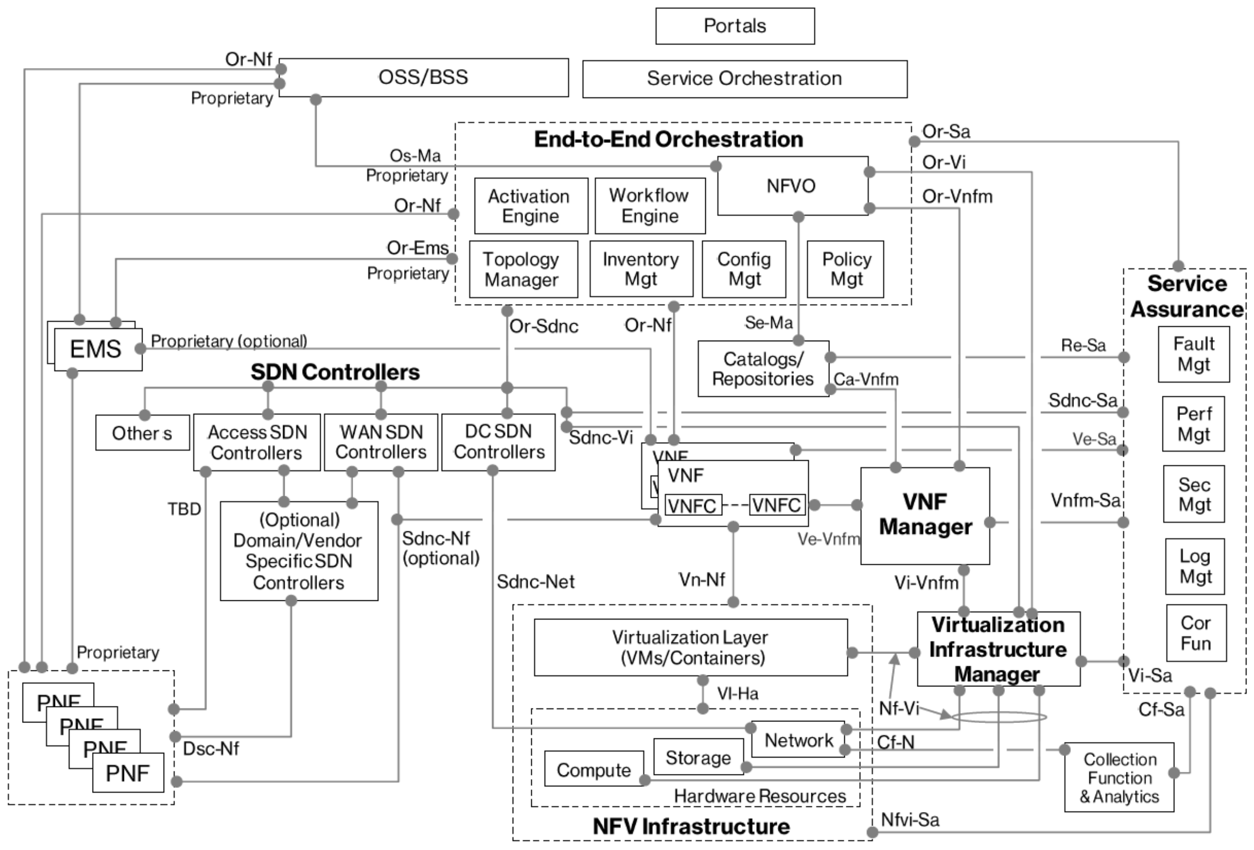

Figure 3 – Verizon SDN-NFV Reference Architecture

This architecture brings together the best of the "pure" ETSI MANO reference framework, the reality of every operator's network, and the market pressure to introduce new network services rapidly. It extends the NFVO-VNFM (datacenter) axis, per ETSI-MANO, with two more dimensions:

- Datacenter - CSP networking dimension, through the introduction of multiple domain controllers (DC SDN, WAN SDN, Access SDN, and Domain Specific controllers), therefore extending ETSI MANO outside of the datacenter;
- Virtual-Physical dimension, where Physical Network Functions (outside or inside the Datacenter) also can be orchestrated from the same 'end-to-end orchestration' functional block.

Further, the Verizon reference architecture allows for the extended configuration management as an example, in addition to EMS-led configuration.

3. Network Orchestration versus Application Orchestration

In the previous chapter, Orchestration is explained as a generic, over-arching concept spanning all networks and all applications as they move to Software (Defined Networking) and (Network Function) Virtualization. Given the vast scope of this transition to SDN and NFV, the industry is tackling the problem from two, essentially orthogonal angles: an application (NFV-led) orchestration problem, and a network (SDN-led) orchestration problem.

As an application orchestration example, consider the deployment of a virtualized mobile core. IMS/VoLTE is a very complex application, consisting of several functions that need to be well-orchestrated through the VNFM. The NFVO function will bring the vEPC and vIMS together into a mobile core, and does this in a static, pre-defined way.

Network orchestration orchestrates several networking functions end-to-end, such that an IP packet flows end-to-end according the rules and policies set out in the network by the orchestrator. This can be a very complex task, as IP underlay and overlay (further defined in Section 4), different domains separated by gateways, physical and virtual appliances and routers may have to be orchestrated to keep the packet flowing. In contrast, the Virtual Network Functions involved in the flow, such as the firewalls, load balancers, WAN optimizers, as applications are typically rather simple “point devices”.

The application and network orchestration approaches currently address very different market demands. Application orchestration focuses on getting the best operational benefits first for otherwise ‘known’ applications such as VoLTE, Network Orchestration focuses on getting the new services enabled by SDN in the market to generate new revenues as soon as possible.

4. Virtual Networks Orchestration and Service Chaining Use Cases

Figure 4 depicts a visual of a Service Chain: a number of Virtual Network Functions, instantiated in the Datacenter as part of the Service Chain through Network Orchestration. In this paper, the authors assume a so-called Overlay Network, VxLAN-based, interconnecting the VNFs through a Software Defined Networking Controller, such as Nuage Networks. The Service Chain further extends the overlay into the WAN domain, where a Network Services Gateway (NSG) connects the Enterprise LAN to the overlay network, transported over the WAN underlay. This Software Defined WAN (SD-WAN) approach allows separating concerns between underlay and overlay, that can support separate administrative domains. The NSG Border Router (NSG-BR) acts as a Datacenter Gateway (DC-GW), bridging underlay and overlay.

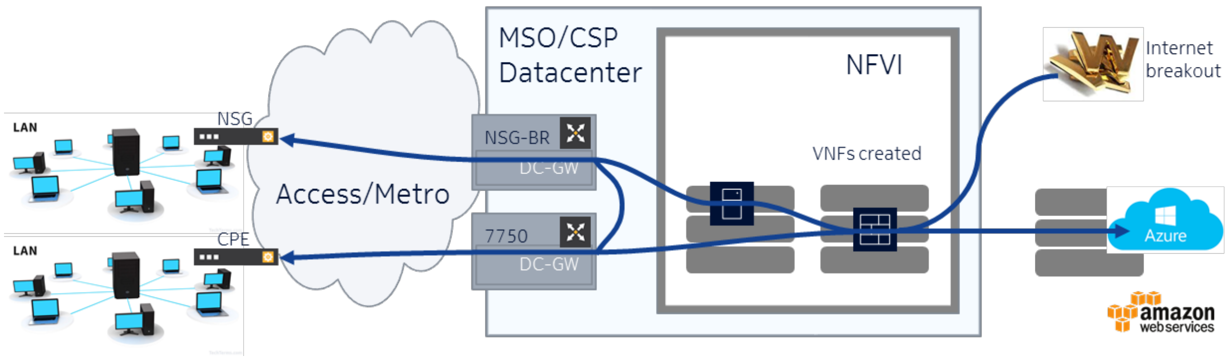

Figure 4 – Enterprise VPN Service Chain example

Service Chains can be further extended into public cloud service providers, such as Amazon Web Services or Microsoft Azure. Interconnection schemes in Virtual Provide Clouds (VPCs) are typically proprietary but well-published and accessible through APIs.

The set up and automation of the Service Chain includes three basic elements of scope:

1. End-to-end overlay connectivity across SD-WAN, the Enterprise Cloud, the CSP's Cloud, and the Public Cloud;
2. Connecting the overlay to the underlay where / when needed;
3. Dynamically include IP appliances in the overlay as VNFs.

This approach is denoted as Virtual Networks Orchestration (VNO): as both Network Functions and Connectivity both together start living in the Cloud, a whole new set of opportunities arise, but at the same time, a new set of challenges come with it.

Several Service Chaining use cases have surfaced in the industry:

- Dynamic Enterprise Services: taking L2 and L3 VPN Enterprise connectivity services to the Cloud, with SD-WAN connectivity allowing for on-net and off-net Enterprise VPN 'instantaneous' overlay connectivity, and with VAS service offerings allowing for upsell. The automated connection of enterprises into the Public Cloud Service Providers and their XaaS products will bring the MSOs and CSPs into the value chain as a trusted middleman, opening the opportunity for MSOs to tap into this one hundred billion dollar market. The discussions further in this paper will use Dynamic Enterprise Services use cases as examples.
- Datacenter Consolidation is a common consideration for organizations that plan to reduce the size of a single facility or merge one or more facilities to reduce overall operating cost and reduce IT footprint. Service Chains typically live across datacenters, and will stream-line the network policies across WAN and datacenters.
- GiLAN Service Chaining is contained in ability to steer traffic from the Gi/SGi interface in EPC through different VAS appliances en route to external networks.
- Virtual Residential Gateway (vRGW) transforms the L3 routed Residential Gateway in the home into bridged RGW, by moving its Layer 3 functions into the CSPs Broadband Network Gateway (BNG). Enriching this capability with features such as Home LAN extension does allow for VAS-in-the-Cloud towards the residential market, such as Cloud Network Attached Storage and Parental Control.

5. The multipliers driving the complexity of Service Chains

Referring again to Figure 4, and specifically for the Dynamic Enterprise Services use case, Virtual Networks Orchestration creates complete end-to-end IP networks. Network Architects do realize the potential complexity of designing end-to-end IP networks, and in addition, these networks need to be instantiated on the fly, and can be short-living. Therefore, automation is key. However, more than 'just automation' is needed (Figure 5):

1. Multiple different classes of end-points for the service chains are possible: Thick CPE, denoting CPEs with VNF hosting capabilities in addition to routing capabilities controlled through Openflow; Slim CPE, with only routing capabilities. Even Thin CPEs are possible, bridged devices terminated onto a DC-GW or DC-I router with vCPE capabilities.
2. Different Enterprises have over time been connected onto the MSO/CSP-managed WAN in evolving ways;

3. The MSO/CSP-managed WAN underlay may have different options (IP/MPLS, …), the MSO/CSP datacenter underbuild may be of different stacks (VMWare, Openstack).
4. At least three different VPC environments will exist in the industry (AWS, Azure, Google Cloud), with a lot of other global and local players (Terramark, OVH, …). Further, each have several options to interconnect at L2 and L3, and are different between public Cloud SPs.
5. In the MSO/CSP-managed datacenter, VNFs will be on the fly instantiated as part of the Service Chain. Also here, different types will be required (Firewalls, Loadbalancers, WAN optimizers, Access Filters, anti-DDoS, …), each dominated by two-three different vendors.
6. On top of all previous multipliers, we have for each different versions, stacking up as time evolves.

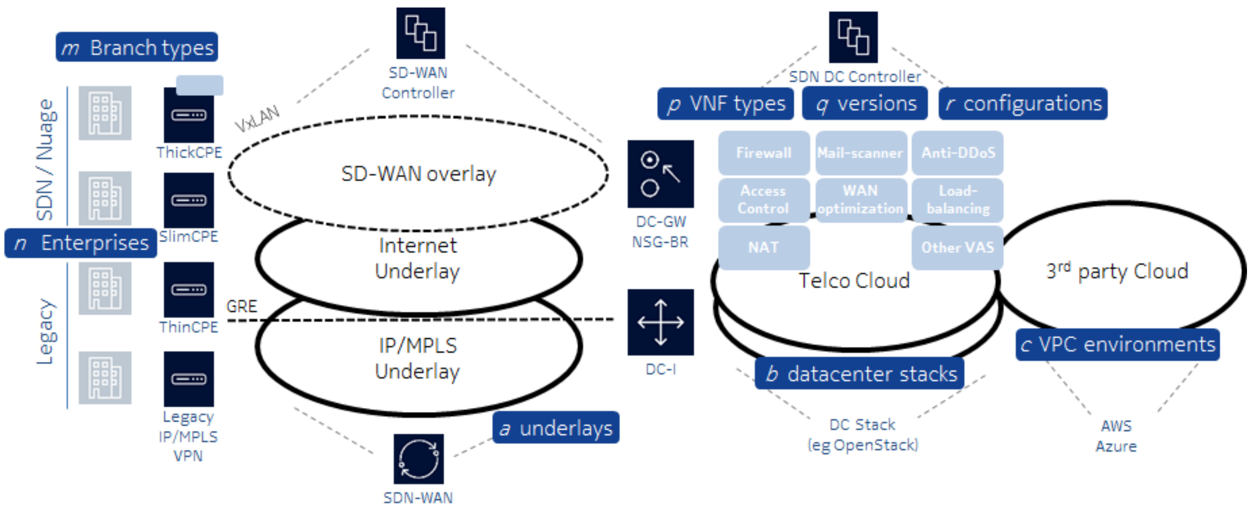

Figure 5 – Enterprise VPN Service Chain in practice: the mulitpliers

In contrast to these multipliers, enterprises do expect to have connectivity ordered through a simple ordering flow, and delivered at cloud speeds – Network as a Service should not be more difficult than other xaaS services.

The implications are clear: Network Services need to be defined and validated in 'a day', for each Enterprise to some extend bespoke, 1000's of which must be served, with service chains at times living only a few hours.

Therefore, in moving end-to-end IP Network Design from the physical world to the virtual world, all variables governing the problem space change, several by orders of magnitude, and a new approach is required to design, develop, test, sell, deploy Network Services, over and over again, in order words, delivering Network Services as a Factory. The association with the concept of a factory is clear: while delivering new services to the market massively and rapidly, the customer still wants to tune the product to his taste, while the end-product delivered should be reliable and cheap.

Network Service Descriptors

The way the Network Orchestration layer is architected is of crucial importance to cope with new and agile end-customer expectations for their network services, while coping with the multiplier-challenge introduced in the previous chapter. With Network Service Descriptors, this becomes a reality. Additionally, this orchestration layer needs to facilitate the open design of new network services by telecom vendors, MSOs/CSPs and/or third-party network service designers, specifically keeping in mind that Network Design is a specific technical skill mastered specifically by Network Architects. The Network Orchestration layer needs to have the capabilities to capture these design requirements of the Network Architects and translate them into the right actions of the SDN/NFV IT systems, preferably without too many, if any translation steps from "High Level Design" specifications, to running software.

The Network Service Descriptors (NSDs) are the blueprint of the overlay and overlay-to-underlay network design as defined in Section 4 above and Figure 4, designed and specified by the Network Architect, and which can be interpreted by the workflow engine to orchestrate the correct service fulfillment and later-on the service assurance.

6. Architecture

The architecture of the Networks Orchestration layer (Figure 6) is designed internally around a workflow engine, controlling the Life-Cycle Management of the Network Service. The data-layer of this workflow engine contains a model of the network service, which can be used to monitor the service, and to destroy, upgrade or transition the service.

The model of this network service is called the Network Service Descriptor (NSD). The NSD is a written declaration of the full network service, described in the TOSCA modeling language. Topology and Orchestration Specification for Cloud Applications (TOSCA) is a language that emerged from the Cloud industry and is specified in YAML, and it describes a topology of cloud based web services, their SDN/NFV components, relationships, and the processes that manage them. The language is standardized by the OASIS and adopted by ETSI/NFV.

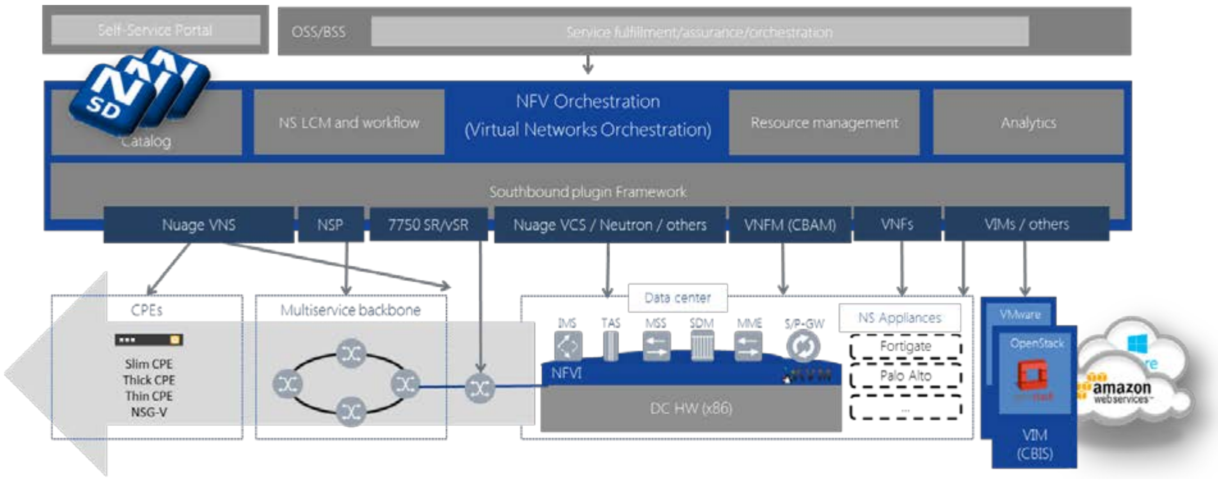

Figure 6 – Architecture of the Virtual Networks Orchestration Layer

Another important component of the architecture is the service catalog. This catalog is a collection of NSDs which are offered, as an example through a self-service portal, as pre-packaged sellable items from which the enterprise customer can select. Through this catalog the provider will be able to offer a large variety of possible network configurations, satisfying the cloud-like service-flexibility requirements of its customers. Additionally, the NSDs themselves will contain a flexible configuration model, which will allow the enterprise customer to personalize its service the way he needs it, but clearly specified by the operator such that the number of variants/customizations are kept under control.

The Southbound plug-in framework is the other component to keep the multiplier under control. This layer will host various plug-ins towards network and cloud equipment. The plug-ins will translate the TOSCA modeled component to real API calls like RestAPI, Netconf/Yang, CLI, … As such this plug-in layer creates the important demarcation layer which separates the concerns between the Network service layer and the real implementation specific configuration needed in the underlying systems (CPEs, Routers, Cloud VIMs/VNFMs, third-party Cloud services, etc).

Last, this layer contains specific resource management and analytics functionalities to ease Root Cause Analysis (RCA), in case the running Network Services experiences run-time issues.

The orchestration layer has an open Northbound interface, which is controlled both from the Self-service Portal as well as from other OSS/BSS components. For those functional blocks the orchestration layer has the important task to hide the dynamic complexity of the underlying Software Defined components (VNFs, NFV, SDNs) triggered by the multiplier explained above, while keeping more limited and static interactions with the upper-layers.

7. Internals of a Network Service Desciptor

The Network Service Descriptor (NSD) is a deployment template which consists of information used by the NFV Orchestrator (NFVO) or Network Orchestrator for life cycle management of a Network Service. The NSD describes the network service at two different layers of abstraction:

- The logical model (Figure 7): describing standardized components such as Virtual Network Functions (VNFs), Physical Network Functions (PNFs), Virtual Links (VLs), Connection Points (CPs), VNF Forwarding Graphs (VNFFGs), Network Forwarding Paths (NFP);
- The implementation model: describing detailed implementation specific components needed to realize the creation of the logical model. E.g. a logical virtual link could be as simple as an Ethernet LAN, or as complex of an overlay VXLAN network mapped on a IP/MPLS underlay configuration.

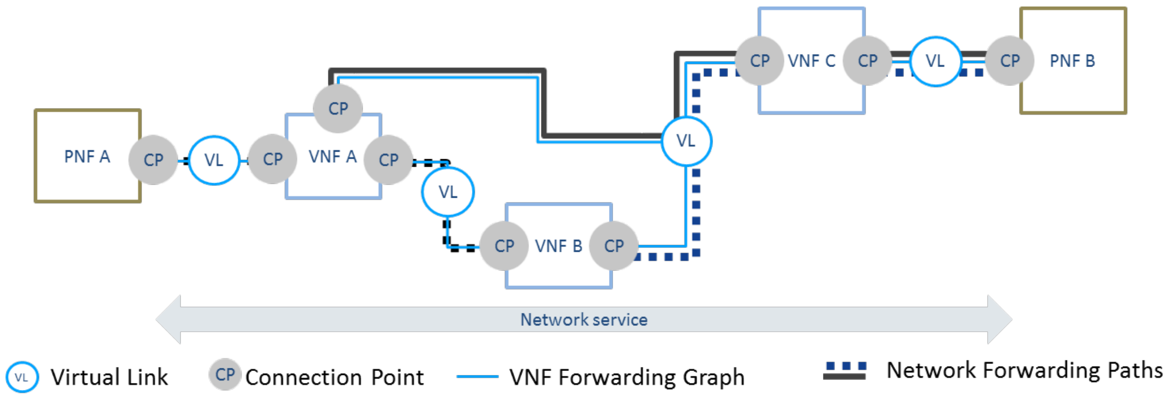

Figure 7 – Logical Model described in the NSD

The components of the logical and implementation models are then written down in YAML notation and packaged together into an archive file, called Cloud Service ARchive (CSAR). This CSAR basically is the packaged network service, which can be onboarded on the service catalog of the orchestration layer.

The YAML description itself contains three major parts: inputs, node_templates, and outputs.

The input part (Figure 8) describes the input variables and represents the flexibility through which the enterprise customer must personalize its networks service: e.g. which IP address ranges to configure, what QoS profile is ordered, whether encryption on the SD-WAN network is needed, …

```
inputs:
    CPE_uplink_QOS_policy:
        type: string
        description: QoS policy to be applied to the uplink port
        default: Silver
        constraints:
            - valid_values: [ Gold , Silver , Bronze ]

    encryption:
        type: string
        description: Select whether encryption between SD-WAN sites is required
        default: DISABLED
        constraints:
            - valid_values: [ DISABLED , ENABLED ]
```

Figure 8 – Example NSD input parameters

The self-service portal can construct automatically out of the input section of the NSD the input forms for the enterprise customer.

The output part (Figure 9) describes the parameters the NSD provides back to the OSS layer. This can either be information for the enterprise customer (such as its public IP address of its Firewall) or information for the OSS to do Assurance monitoring.

```
outputs:
    fw_mgmt_ip:
        value: { get_attribute: [ fw_vnf, outputs, [ fw_mgmt_ip ] ] }
    fw_public_ip:
        value: { get_attribute: [ fw_vnf, outputs, [ fw_public_ip ] ] }
```

Figure 9 – Example NSD output parameters

The main body of the TOSCA files describes the node_templates. These are a collection of components which describe all the various items which require configuration, together with the properties of this configuration. The important difference with other automatization languages like Ansible and YANG is that complex relations and dependencies can be described between these components. This makes it possible to interpret this description by the workflow engine, in order to make a perfect, well sequenced service fulfillment possible.

```
pub_subnet:
    type: nokia.nuage.nodes.Subnet
    properties:
        name: Public_Internet
        associatedSharedNetworkResourceID: { get_attribute: [ shared_internet , ID ]}
    requirements:
        in_zone:
            type: nokia.nuage.relationships.DefinedInZone
            target: dc_public_zone
        shared_network:
            type: tosca.relationships.DependsOn
            target: shared_internet
```

Figure 10 – Example of a node_template

Given this explanation one could see the parallel between this Network Service Descriptor and the Apps that are installed on smartphones. For the enterprise customer, installing a new network service or upgrading an existing one, will be as simple as instantiating one of the NSDs offered by the 'webshop' of the MSO/CSP. The NSD contains all description of the flexibility the service has and at runtime contains all information about that service. This contained modeled information of the network services makes it therefore possible to automatically do advanced life-cycle actions, such as upgrading the service to a new version or transiting the service from the enterprise from one SD-WAN service (eg, a hub and spoke SD-WAN configuration) to a complex other network service (e.g. a full mesh SD-WAN with Firewalls and Intrusion Detection VNFs). By analyzing the dependency graphs, the orchestration layer exactly knows which actions to automatically complete. All of these described features are inherited from the interesting capabilities offered by the TOSCA description language in which the NSDs are written.

Of course, TOSCA has some drawbacks such as limited set of standardized intrinsic functions and relative static topology of the dependency graph. These drawbacks can be overcome by introducing more intelligence in the plug-ins and making a smart NSD decomposition. It is expected that the TOSCA standard will further evolve to onboard these requirements.

When putting all blocks together, one can now fully describe the NSD 'path' within the Orchestration layer (Figure 11). The NSD describes all configuration items to be done, together with the relationships between them. The workflow engine in the orchestration layer analyses this dependency graph and uses the plug-ins to do all the configurations required across the various network elements and cloud platforms.

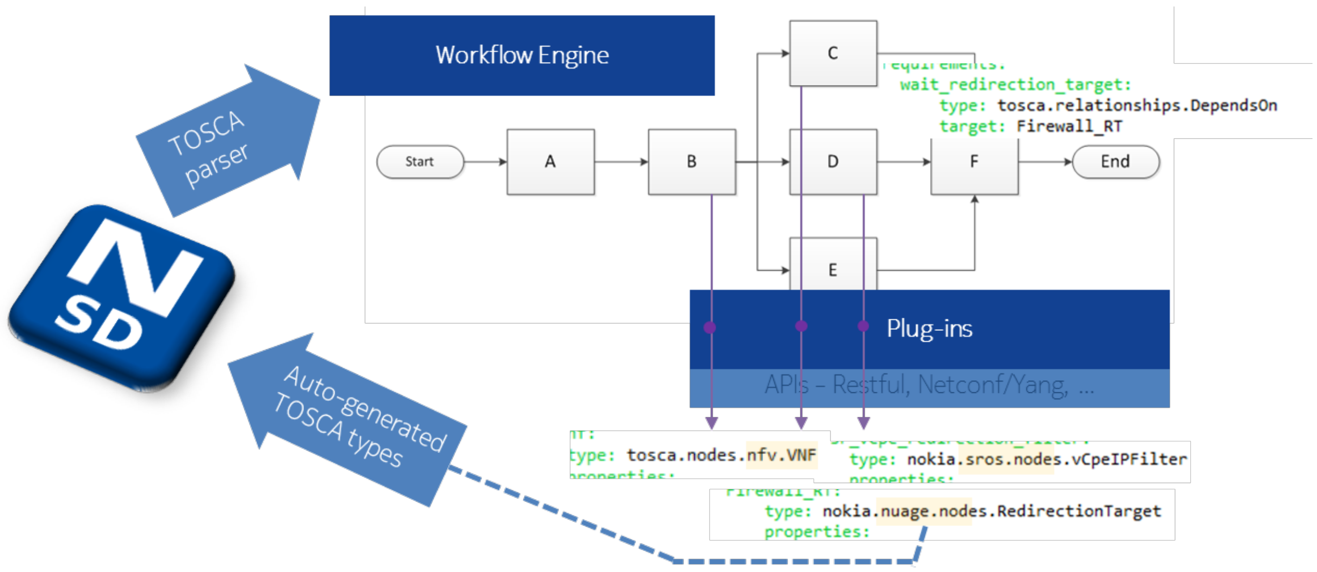
Workflow Engine
TOSCA parser
Start
A
B
C
D
E
F
End
wait_redirection_target:
type: tosca.relationships.DependsOn
target: Firewall_RT
Plug-ins
APIs – Restful, Netconf/Yang, …
Auto-generated TOSCA types
type: tosca.nodes.nfv.VNF
type: nokia.sros.nodes.vCpeIPFilter
type: nokia.nuage.nodes.RedirectionTarget
properties:

Figure 11 – An NSD into action

The most interesting thing however is that also automation can be realized in the generation of these node_templates and plug-ins. If clearly described interface specifications are available (such as OpenAPI, YANG models, …) both plug-ins and node-templates can be automatically generated and updated, such that the versioning factor of the earlier described multiplier-challenge is overcome.

‘Best Practice’-based Abstraction

The previous chapter described how one NSD can be written and executed. However, the full end-to-end service, as it is deployed by the Service Orchestration layer, typically consists of a combination of multiple NSDs. As example, an enterprise might initially start from a SD-WAN service with a number of branch-offices. Over a period of time, the enterprise will typically add and delete branch-offices, and might subscribe to new managed services inside the data-center (e.g. firewall, intrusion detection, spam filters,…).

Besides the pure technical skills of writing a NSD, business requirements will highly influence which configuration actions will be grouped together into one ‘sellable item’ – from MSO/CSP to the enterprise customer (adding a ‘branch’ will require the enterprise to pay an additional subscription fee), and which ones will best be decomposed into complementary NSDs (a ‘branch’ hence to become a separate NSD). The following business requirements will highly influence the decomposition of the end-to-end service into NSDs:

- How will the MSO/CSP allow the enterprise-customer to mash-up combinations of service-components in order to personalize its configuration?
- Which migration, upgrade or up-sell scenarios to new functionality would the MSO/CSP like to foresee?

Based on those Best-Practices we currently see already the following collections of separate NSD families (Figure 12), which will together create an end-to-end Network Service.

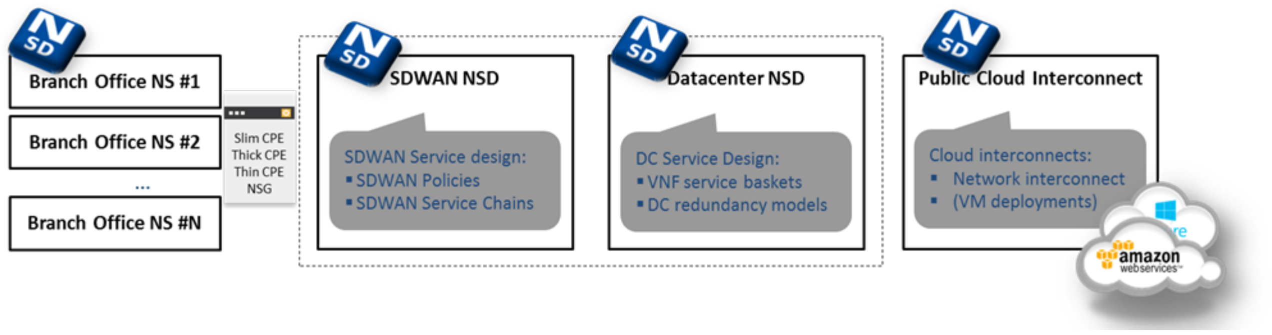

Figure 12 – NSD best-practice decomposition. Example for Dynamic Enterprise Services.

- SD-WAN NSDs will typically contain the model on how an enterprise would like to interconnect its branch offices. This can be a full-mesh scenario, a hub and spoke scenario (where the hub is e.g. the IT headquarter of the company), etc. In addition, it will contain pre-configured configurations of QoS policies, ACLs and Application Aware routing scenarios.
- Branch Office NSDs will typically contain various flavors of branch office interconnectivity. E.g. whether the branch office is connected via single uplink or dual uplink, whether a SDN enabled CPE (so-called ‘SlimCPE’) is used, a ThickCPE - with capabilities to run VNFs distributed between cloud and branch office, or whether a so-called ThinCPE configuration is deployed, which interconnects a non-SDN legacy router to the Overlay SD-WAN.
- Datacenter NSDs will typically contain the needed interconnect and service chaining between the SD-WAN and the Managed Services (VNFs) in the telco-cloud. These NSDs might also contain knowledge of Active-standby configurations when service chains are defined across multiple redundant datacenters.
- Other independent NSDs such as interconnects to various third-party public cloud environments (AWS, Azure,…), interconnects to fixed/wireless infrastructures (like GiLAN, vResidential GWs,…), or complementary service packs (like advanced application monitoring through Application Aware Routing,…)

This best-practice decomposition provides high flexibility to the enterprise customer for defining its required service, and will create a higher-level abstraction towards the Service Orchestration layer, such that the details of the Network definitions are handled in the Network Orchestration layers while the Service Orchestration layer instantiates actions on billable items requested by the end-customer.

Intent-based testing

Testing represents an important aspect of the NSD development process, meant to ensure that a deployed NS performs per the business requirements. This aspect takes center-stage when the development process adheres to modern DevOps or Continuous Delivery methodologies, which is the case of the Network Services as a Factory concept.

Considering the short ‘one-day’ cycles imposed to NS development and validation, it becomes imperative to automatically execute the defined tests every time a change is committed to the NSD implementation. Given the declarative nature of the TOSCA domain-specific language, the most appropriate test methodology is also a declarative one, in which the NS state and behavior is validated against a selected set of inputs.

The test methodology defining the 'Network Services as a Factory' approach embraces the notion of behavioral driven testing which checks that the purpose the NSD was designed to fulfill, is successfully achieved. To bridge the gap between business requirements and test use case implementation while retaining full automation capabilities, the authors have selected the Cucumber framework due to its capability of supporting the use of natural language in test definitions.

The domain-specific natural language subset implemented by the chosen test methodology covers the complete creation and deployment lifecycle of a network service according to ETSI standards and is open to further extensions. The below code snippet (Figure 13) shows an example of testing the Internet connectivity established via a data center service chain connected to a full-mesh SD-WAN VPN. An important note is that the best practice related to scenario autonomy in Cucumber was deliberately broken in order to allow (a) the required flexibility in defining large-scale test use cases and (b) avoid the unnecessary repetition of expensive provisioning scenarios.

```
Feature: Connect a DC service chain to an existing Nuage VPN
    In order to provide E2E connectivity
    As a Network Service Designer
    I want to define an DC service chain
    That connects to an existing Nuage VPN
    And provides Internet access to the sites connected by it

    Background:
        When I log in to CBND

    Scenario: Model the existing VPN as a mesh
        When NSD "../AT_csar/Site-to-site_Mesh_connectivity.zip" is uploaded to CBND with alias "NSD Mesh"
         And NS instance "AT Mesh VPN" of alias "NSD Mesh" is created with parameters:
            | enterprise_name | Enterprise_Test |
            | encryption      | DISABLED |
            | domain_name     | AT_SD-WAN_Domain |
         And NS "AT Mesh VPN" is deployed within 15 min
        Then NS "AT Mesh VPN" state should be DEPLOYED
...
    Scenario: Site-2-site and site-2-outside traffic are enabled
       Given NS "AT Branch 1" state is DEPLOYED
         And NS "AT Branch 2" state is DEPLOYED
         And NS "AT DC SC" state is DEPLOYED
        Then below VMs can pass traffic to each other
            | Name     | Interface |
            | g1lanvm4 | eth2      |
            | g1lanvm5 | eth2      |
         And VM "g1lanvm4" can access URL "www.google.com"
         And VM "g1lanvm5" can access URL "www.google.com"
```

Figure 13 – Intent-based testing – a code snippet.

This solution allows the NS developer or designer to quickly define test use cases that are versioned using a source code management (SCM) system and automatically executed by a continuous integration (CI) server against a specified deployment environment, as shown in Figure 14. The presented Network Service as a Factory implementation relies on the NodeJS implementation of Cucumber - cucumber.js - which runs inside a Jenkins CI server.

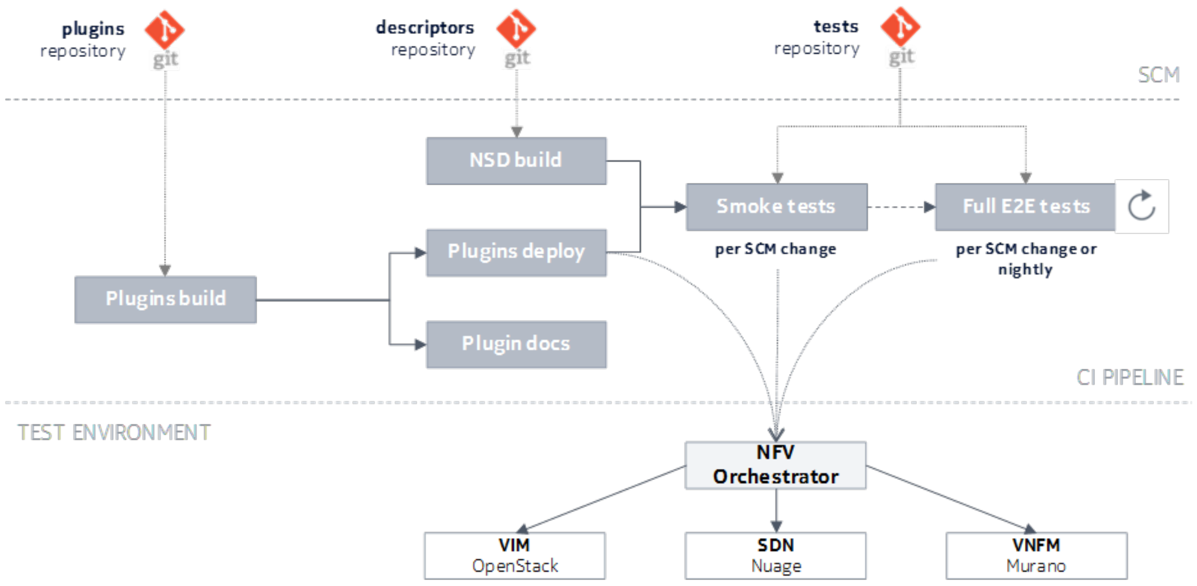

Figure 14 – NSD testing as part of the Continuous Deployment pipeline.

The advantages brought in by the presented NSD testing methodology are multiple. Some of them stem from its alignment to the Continuous Delivery paradigm while others explicitly address the specifics of the NSD design and development process.

- The fully automated nature of the process ensures a continuous, fast and objective assessment of the NSD quality. The network service designer can stay focused on describing the higher-level NSD structure and its intended behavior, knowing that the CI platform takes over the validation efforts. Given that NS designers usually align with the network engineer profile more than with the software developer one, this represents a significant enabler for network engineering teams in transitioning to the Network Service as a Factory approach.
- Every change brought to one element in the toolchain, be it an NFV orchestrator plugin, an NSD or a test definition, will be automatically deployed and validated on the target environment in the context of an end-to-end solution. This keeps quality high, reduces validation times and removes the risk of regressions and their associated high costs. Given NFVO plugin development usually requires technologies different than NSD development, the presented approach bridges the gap between the different teams handling these activities by bringing them under the same validation and acceptance umbrella.
- The use of natural language in test definitions streamlines the customer acceptance process since the test results can be interpreted by all project stakeholders as opposed to just the technical teams. Furthermore, since the platform can be directed at different environments, it is perfectly possible to use it to deploy and validate NSDs on a new customer environment with positive impact on project start-up costs and delivery timelines.
- Besides the abstraction towards the OSS layer brought by the NSD concept as described in the previous chapters, behavioral-driven test definition facilitates the integration of new network services with the OSS by giving OSS teams insight into how the service should be managed and what are its requirements without exposing the complexity of its implementation.

Conclusion

While Network Orchestration and Application Orchestration share technologies coming from the domains of SDN and NFV, the two approaches serve different objectives. The paper has specifically addressed Network Orchestration, aimed at launching new revenue generating Business Connectivity services rapidly, with Value Added services running in private or public cloud data centers chained-in into the Network Service.

The rapid market traction of SD-WAN overlay IP networking, enabled by product suites such as Nuage, fuels the requirements for Network Orchestration: overlay to underlay connectivity, connectivity into Public Cloud, Internet Break-out, flexible onboarding of virtualized Value Added services such as firewalls, wan optimizers, email filters, and others, all easily customizable for that one specific enterprise customer of the MSO/CSP. However, the combination of different Value Added Services vendors and types, SDN domain controllers, cloud stacks, software versions, implies a multiplier in the number of combinations that are possible to deliver just that one single end-to-end service chain over and over again. A different approach is needed, bridging the need for instantaneous Network Service delivery, and network diversity, complexity, and constant evolution.

A combination of technologies, approaches and methodologies, building on abstraction, automation, and decomposition, are brought together, in an approach called Virtual Networks Orchestration. It allows to deliver Network Services and NSDs rapidly, massively, and reliably – Network Services as a Factory.

TOSCA as a YAML-based Domain Specific Language allows to describe Network Services in Network Service Descriptors. More specifically, a best-practice based Network Service decomposition approach allows to abstract out the different areas of complexity in the network, so that, through isolation, a change in the service description in one area does not require to change the full network service. 'Best-practice' should be based on the interaction of the market facing units of MSOs/CSPs with its customers (enterprises as examples).

Intent-based testing is explored as a further step towards automation. To bridge the gap between business requirements and test use case implementation while retaining full automation capabilities, the Cucumber framework was highlighted, and its capability of supporting the use of natural language in test definitions. The advantages brought in by the presented NSD testing methodology include the continuous, fast and objective assessment of the NSD quality; the validation of a change to any of the elements in the toolchain, NSDs and plug-ins; the streamlining of the customer acceptance process; and facilitating the integration of new network services with the OSS.

Abbreviations

CI	continuous integration
CLI	command line interface
CSP	communications service provider
DES	dynamic enterprise services
DSL	domain-specific language
EMS	element management system
MANO	management and orchestration
MEF	metro ethernet forum
MSO	multiple systems operators
NFV	network function virtualization
NFVO	NFVorchestrator
NS	network service
NSD	network service descriptor
OASIS	organization for the advancement of structured information standards
ONF	open networking foundation
OSS	operations support systems
SDN	software-defined networking
SD-WAN	software-defined wide-area networking
Tosca	topology and orchestration specification for cloud applications
vePC	virtualized enhanced packet core
vIMS	virtualized IMS
VNF	virtual network function
VNFM	virtual network function manager
VoLTE	voice over LTE
VPC	virtual private cloud
VPN	virtual private networking
xaaS	infrastructure, platform, software as a service

Bibliography & References

http://innovation.verizon.com/content/dam/vic/PDF/Verizon_SDN-NFV_Reference_Architecture.pdf

http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf
http://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/014/02.01.01_60/gs_NFV-IFA014v020101p.pdf

https://www.oasis-open.org/committees/tosca/
http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/tosca-nfv-v1.0.pdf
http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/os/TOSCA-Simple-Profile-YAML-v1.0-os.pdf
http://yaml.org/

http://www.nuagenetworks.net/

Getting Grounded with DevOps, 2015, https://www.hpe.com/h20195/V2/getpdf.aspx/4AA4-3696ENW.pdf

Continuous Delivery: Automating the Deployment Pipeline, 2016, https://www.microfocus.com/media/white-paper/continuous_delivery_automating_the_deployment_pipeline_wp.pdf

Hallenberg N. and Carlsen P.L., Declarative automated test, Proceedings of the 7th International Workshop on Automation of Software Test, pp. 96-102, 2012

Crowther M., An Introduction to Behavioral Driven Testing, 2009, http://www.cyreath.co.uk/papers/Cyreath_An_Introduction_to_BDT.pdf

Cucumber reference, 2017, https://cucumber.io/docs/reference
Cucumber Best Practices, 2015, https://github.com/strongqa/howitzer/wiki/Cucumber-Best-Practices