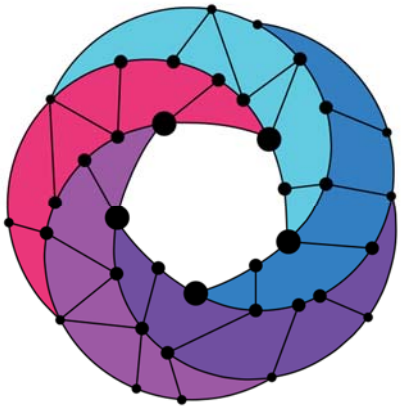

Open Source MANO

EXPERIENCE WITH NFV ARCHITECTURE, INTERFACES, AND INFORMATION MODELS

A White Paper prepared by the OSM End User Advisory Group

Issue 1
May 2018

ETSI (European Telecommunications Standards Institute)
06921 Sophia Antipolis CEDEX, France
Tel +33 4 92 94 42 00
info@etsi.org
www.etsi.org

Authors

Milind Bhagwat	BT	
Don Clarke	CableLabs	
Phil Eardley	BT	
Antonio Elizondo Armengol	Telefónica	
Gerardo García de Blas	Telefónica	
Pål Gronsund	Telenor	
Adrian Hoban	Intel	
Serge Manning	Sprint	
Tetsuya Nakamura	CableLabs	
Francisco Javier Ramón Salguero	Telefónica	
Andy Reid	BT	(Editor)

Contents

1 Introduction and Summary

In this paper we discuss the lessons we have learnt as an open source community in the implementation and adoption of the NFV specifications developed by the ETSI NFV ISG. This paper is the first of a number of potential papers that OSM community intend to produce which discuss the experience of the OSM in achieving compatibility with the work of other bodies

The NFV standards are necessarily complex and assessing interoperability according to these standards is also complex. However, the rise of open source implementation projects gives a fresh approach to achieving effective interoperability.

OSM set out 24 months ago with the aim of providing the industry with a fully functional orchestrator for NFV implemented as open source and aligned to the ETSI NFV framework. The use of the ETSI NFV standards was the starting point and close work between OSM and the ETSI NFV ISG has always been part of the OSM plan. OSM made available Release FOUR which reaches a high level of maturity both in the supported features and in the robustness of the code. At this point, we reflect on a number of lessons we have learnt in reaching this stage.

Indeed, one of the main objectives of OSM from the beginning was "learning by doing" with a full recognition that NFV orchestration is breaking new ground for interoperable systems and the OSM End User Advisory Group (EUAG) set this out[1]. We have all acknowledged both the lack of precedents from which to draw learning and the complexity of the interface standards themselves.

As illustrated in Figure 1 below, in contrast to the one-way 'waterfall' process with proprietary developments by vendors, which is the norm in many communications service providers' infrastructures today, open source gives the possibility of achieving interoperability for service providers by an open cyclical process. There are many benefits to this process for developing the complex systems needs for services providers, not least for the implementation of 5G networks.

[1] For example, at the SDN/NFV World Congress, November 2017, in The Hague: https://osm.etsi.org/wikipub/index.php/OSM_workshops_and_events#OSM_Workshop_.40_SDN_.26_NFV_World_Congress_2017.2C_The_Hague_.28Oct_9.2C_2017.29

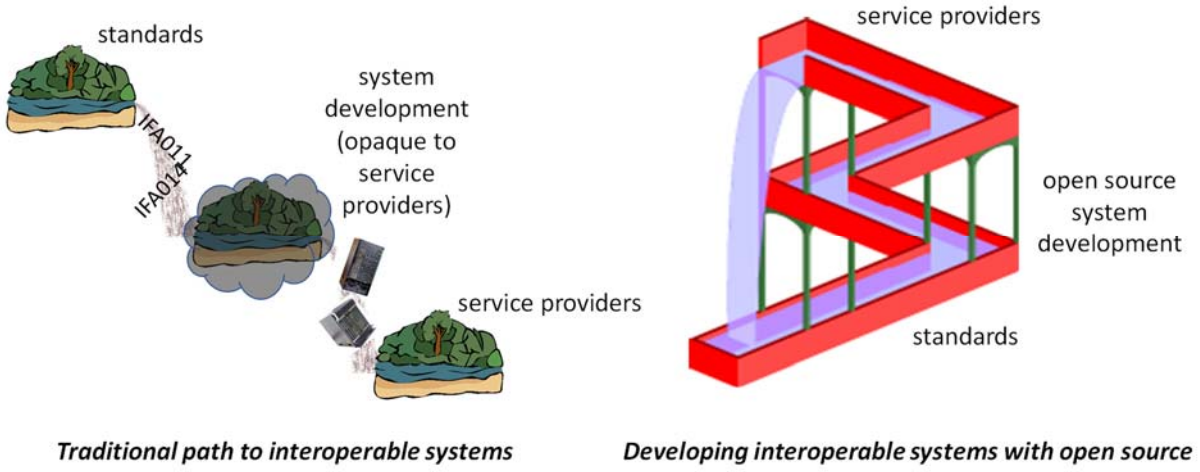

Figure 1 the opportunity of open cyclical development to achieve interoperability

With complex standards, some level of ambiguity or functional deficiency might be expected, even with the best possible intentions of those involved. However, in any standards setting process, there are inevitable differences of technical opinion, commercial interests, and other conflicts which can result in compromises to the integrity of the standards. They may not be apparent, even to those directly involved, and there should be no reason to believe that the ETSI NFV ISG standards should be uniquely immune from this. OSM provides the opportunity through its open development of a real system to validate these standards and provide open and early feedback.

This white paper is structured in the following sections and the summary conclusions of each section are included below.

- Section 2 outlines the ETSI NFV ISG standards and positions of OSM with respect to them, given the implementation options that OSM has made.
    - OSM takes the VNFM and NFVO functional blocks together and re-architects them as different internal modules which, as a whole, offer the same functionality and expose the same reference points. As one of the benefits of this approach, OSM does not need to implement any VNFM to NFVO granting/reservation functionality.
    - From the architectural options set out in IFA009 for VNFM, OSM chooses the architectural decision to implement the generic VNFM approach, and provides all the elements needed to avoid the need to require a multi-VNFM option in practice. In addition, there is an architectural path to integrate with specific VNFMs if required
    - OSM is agnostic on the implementation of the Ve-Vnfm-em and Ve-Vnfm-vnf reference points. In practice, OSM provides an isolated execution environment for a configuration, management and control agent for a VNF. This means that the actual interaction between OSM and the VNF is mediated by a dedicated agent whose code is provided by the VNF vendor as part of the VNF package.
    - OSM augments the NFVO role with service orchestration capabilities that include the ability to add day-2 configuration primitives to manage NS instances once deployed.

- Section 3 discusses all the various aspects of the ETSI NFV work relevant to OSM and reviews the extent to which OSM has been able to follow the results of different working groups. In most areas, this has been straightforward, however, we note the following points:
    - The outputs of MANO and PER WGs in Phase 1, and EVE (evolution), TST (test), REL (reliability), and SEC (security) WGs in Phase 2 have been clear and sufficient to allow OSM to align to the outputs of those working groups.
    - In Phase 2, the output of IFA WG, by its own nature, has been very extensive and complex. Unsurprisingly, its application from the perspective of the implementation could not be considered straightforward or even useful in a number of aspects, particularly from the perspective of an accurate description of the resources or application lifecycle. These issues arising from the output of IFA are the main subject of Section 4.
    - The output from SOL is at a much earlier stage and OSM is now taking it as one of the key building blocks of its North Bound Interface (NBI).
- Section 4 makes a number of specific observations on the interface specifications and interoperability based on the experience of OSM to date.
    - The section starts with a general observation that successful automation across an interface requires the interface specification to be *parsimonious*, that is to say, it specifies only what is necessary and all that is sufficient.
    - Specific observations associated with descriptors and on-boarding
        - The OSM information model in based on the NFV information model for the VNFDs and NSD defined in IFA011/IFA014 with additions to support explicit specification and control some features of the infrastructure where the IFA011/IFA014 specification is overly abstract for the needs of OSM, notably EPA features.
    - Specific observations associated with southbound interfaces
        - Looking south, OSM consumes VIM and SDN controller APIs as they are. The understanding inherent in the IFA005 standard has been helpful in developing OSM's internal resource abstraction model, although it must be noted that to date no VIM or SDN controller has directly implemented IFA005/IFA006, so OSM decided to put in place a plugin model to support real VIMs and SDN controllers that exist today.
    - Specific observations associated with northbound interfaces
        - Looking north, current work on the OSM northbound API suggests that further work would be helpful to get closer to a parsimonious specification.
        - In contrast to the control interfaces for a VNF, which presents the VNF as a single coherent entity, the NFV ISG specifications present the control of a NS as a set of VNF and VL components. This conservative approach, while valid as an option, can become a barrier for effective automation as it does not help to reduce complexity, still requiring the OSS to play a central role in the control of the internal structure of the NS.
        - In order to address this issue, OSM allows the definition of NS-related primitives in the NS Package so that it is possible to control the NS as a whole and drive actions over the NS as if it were a single object.

- Section 5 outlines some consequences for standards bodies, other open source projects, services providers, vendors, and academic institutions. Specific challenges identified include the following:
    - There would be great value in extending and developing more detail within the NFV architecture. In particular, it would be helpful in reducing ambiguity if the architecture a) separately described logical functionality and domains of implementation, b) provided a clear way by which 'as a service' can be consumed by NFV, c) developed clearer end to end automation processes.
    - The NFV requirements have some subtle but important differences from cloud requirements, arising from the fact that generally cloud applications are 'end systems' while VNFs generally sit 'in the wire'. This means that VNFs and NSs, compared to cloud applications, tend to have different resource bottlenecks and to have more than one network interface. This means that making simple reuse of cloud systems has significant challenges.
    - Whilst automated conformance testing is facilitated by the use of frameworks such as REST and YANG, these automated tests are not the same as interoperability testing and are most unlikely to be sufficient to guarantee interoperability at the end-to-end system level. Whilst these frameworks are helpful, achieving full interoperability still remains a challenge.
    - Some of the specifications involve many layers of specification and different standards bodies may be responsible for different layers. This presents a challenge to making enhancements to an overall specification which are consistent and coordinated.
    - By employing an open development process, open source invites a level of engagement from carriers and end users at the component and systems integration level, possibly at a much greater level than had been the case with proprietary developments.

2 Positioning OSM Release FOUR in the NFV Architecture

2.1 ETSI NFV ISG Documentation

The general NFV architecture was set out in the original four specifications published by the ETSI NFV ISG in October 2013 and the architecture diagram (Figure 4 of GS NFV 002) has been a basic blueprint ever since. Currently, the ETSI NFV ISG has a comprehensive set of semantic level specifications covering each of the reference points identified in the basic architecture (from the IFA working group). In addition, there are now a number of detailed syntactically precise specifications for some of the interfaces (from the SOL working group).

The mapping of these documents is usefully shown in Figure 2 below, which is reproduced from the ETSI NFV web site.

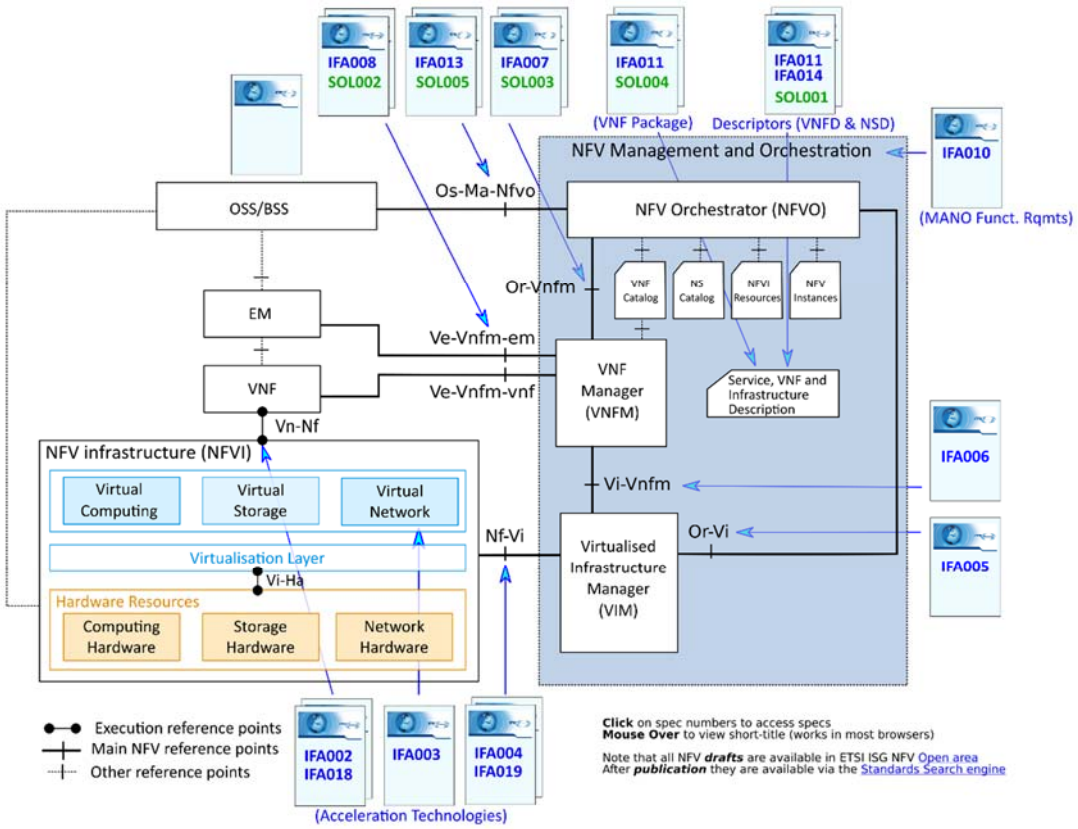

Figure 2 Mapping of IFA and SOL specifications to the reference points of the NFV architecture.

In addition to these current documents, there is an earlier ETSI NFV document for management and orchestration, NFV MAN001.

The following interface specifications are therefore of direct interest to OSM:

	Semantic Specification	Syntactic Specification	Earlier Specification
Or-Vi	NFV IFA005	-	NFV MAN001
Vi-Vnfm	NFV IFA006	-	NFV MAN001
Or-Vnfm	NFV IFA007	NFV SOL003	NFV MAN001
Ve-Vnfm-em	NFV IFA008	NFV SOL002	NFV MAN001
Ve-Vnfm-vnf	NFV IFA008	NFV SOL002	NFV MAN001
Os-Ma-Nfvo	NFV IFA013	NFV SOL005	NFV MAN001
NSD	NFV IFA014	NFV SOL001	NFV MAN001
VNF Package	NFV IFA011	NFV SOL004	NFV MAN001
VNFD	NFV IFA011	NFV SOL001	NFV MAN001

Table 1 ETSI NFV ISG interface specifications of relevance to OSM

As well as these reference point specifications, NFV IFA010 sets out the functional requirements[2] of the NFVO, the VNFM, and the VIM respectively whilst NFV IFA009 discusses different options for the implementation of these functions and in particular covering different implementation options for the VNFM.

The reference point specifications are organized as a set of specific interfaces offered on each side of the reference point. Figure 3 illustrates these interfaces and where they are offered.

Three of the reference points (Or-Vi, Vi-Vnfm, and Os-Ma-Nfvo) are true client/server relationships in that all the interfaces are offered on only one side of the reference point and consumed by the other side. The consuming side offers no interfaces (shown in the figure with the empty set symbol, Ø).

[2] Functional requirements in the context of IFA010 is more about functional calls on an interface rather than about functional blocks. For example, IFA010 does not give a functional decomposition of the main MANO functional blocks.

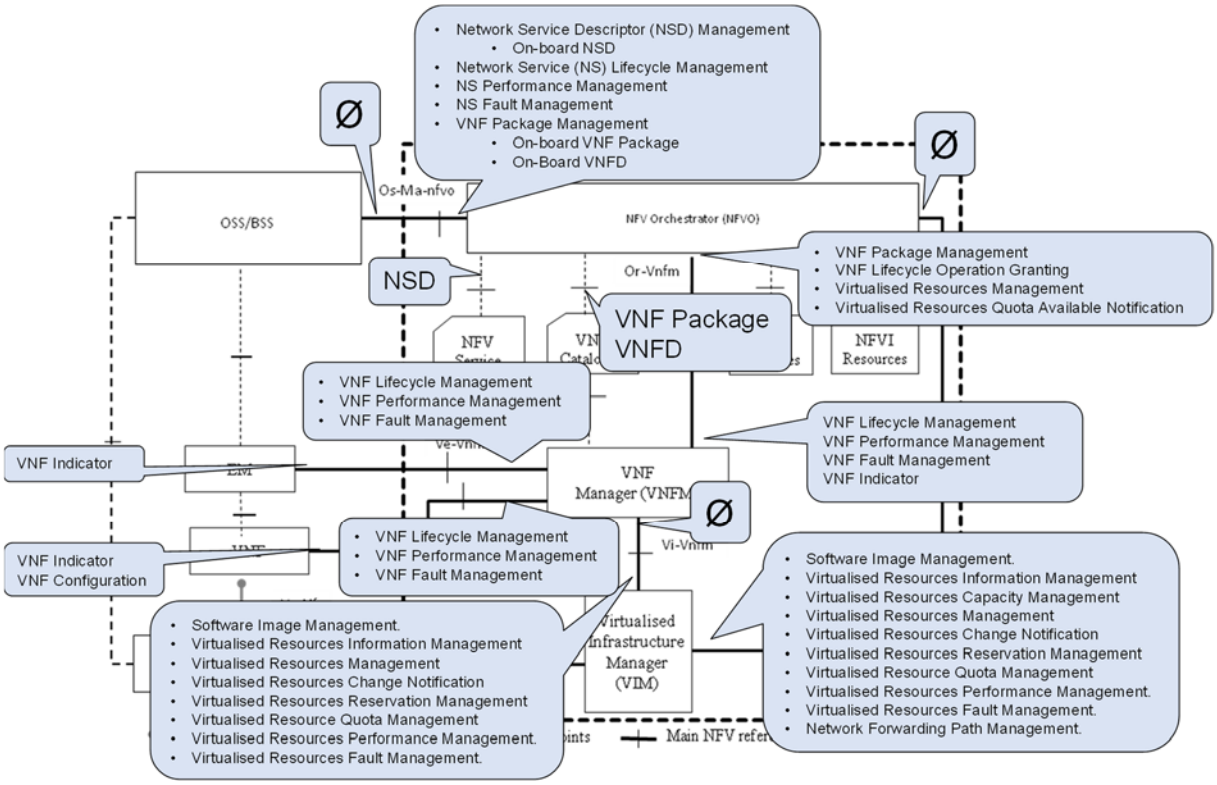

Figure 3 Specific interfaces offered on each side of each reference point.

2.2 OSM Choices Under GS NFV IFA009

OSM made its particular choices from the options set out in IFA009, specifically, OSM made the architecture decision to:

- implement a generic VNFM as an integral component of OSM;
- split its functionality into submodules which are focused on specific tasks and provide internal abstractions for particularly elaborated procedures. Thus, OSM has a Resource Orchestration (RO) component to handle cloud and physical resources, a VNF Configuration Agent (VCA) component to handle the interaction with VNFs/applications, and a monitoring component (MON) module in charge of monitoring procedures, on top of which sits a layer of Service Orchestration (SO), in charge of providing E2E coherency.

These option choices give a broad mapping between the main OSM components and the NFV functional blocks as illustrated in Figure 4.

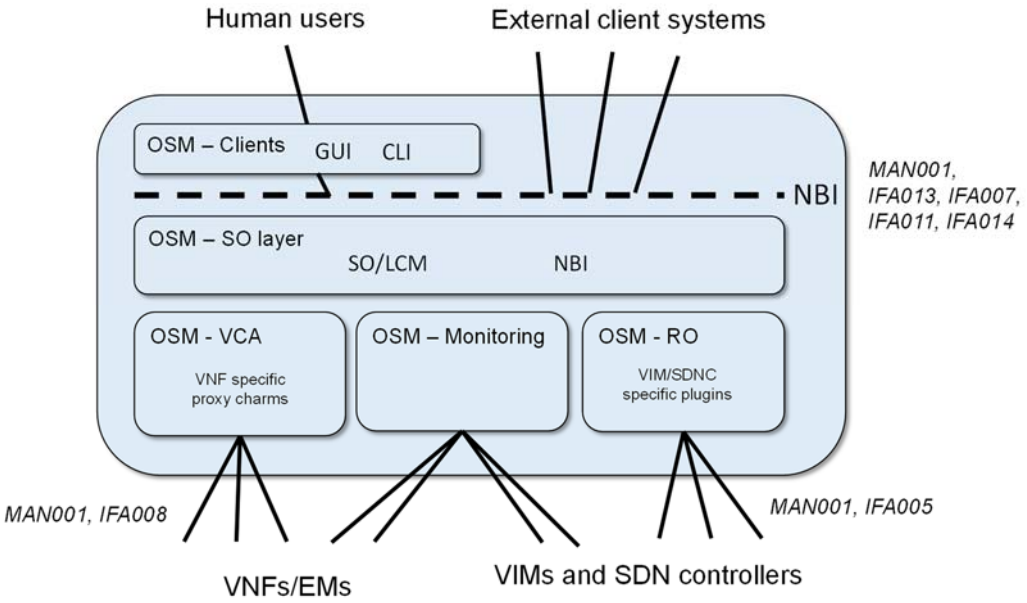

Figure 4 Broad mapping of OSM components to NFV functional blocks resulting from OSM choices under IFA009

OSM recognises that some of the interfaces in the IFA specifications exist to support different options for implementations of VNFMs. By taking the decision to include the functionality of generic VNFM within OSM and to split the modules internally to seek better software specialization, many of the interfaces of Figure 3 are not required by OSM. In particular, all the interfacing associated with passing control of resource assignment by grants and reservations between the NFVO and the VNFM on the Or-Vnfm reference point is not required neither the entirety of the Vi-Vnfm reference interfacing, at least not differentiated from Or-Vi. Thus, all the interactions with the VIM are handled in a single interface which collapses Or-Vnfm into Or-Vi.

On the other hand, the generic VNFM implemented by OSM does not require the VNF to offer a specific type of interface. On the contrary, **OSM can consume whatever particular interface is offered by a VNF and/or its EM** by using VNF-specific code plug-ins, realised as proxy charms —ready to run in a container in the VCA— and which come as part of the corresponding VNF package. Thus, OSM can consume any of the interfaces which are currently presented by commercial VNFs — whether directly, via an EM, or both—, acknowledging the current reality of multiple configuration methods that can be found nowadays in NFV and cloud environments.

In essence, the VCA can be seen as performing a mediation role between a common and abstract VNFM and the wide variety of specific VNF types in the same way that the RO mediates between the common and abstract VNFM and the wide variety of specific types of VIMs and SDN controllers.

With the choices OSM has made under the options discussed in IFA009, Figure 5 shows the interfaces that OSM directly supports. On the other hand, Figure 6 shows the interfaces that OSM does not need to support externally either because they are not relevant to the OSM choices or because they are internalised inside the OSM system.

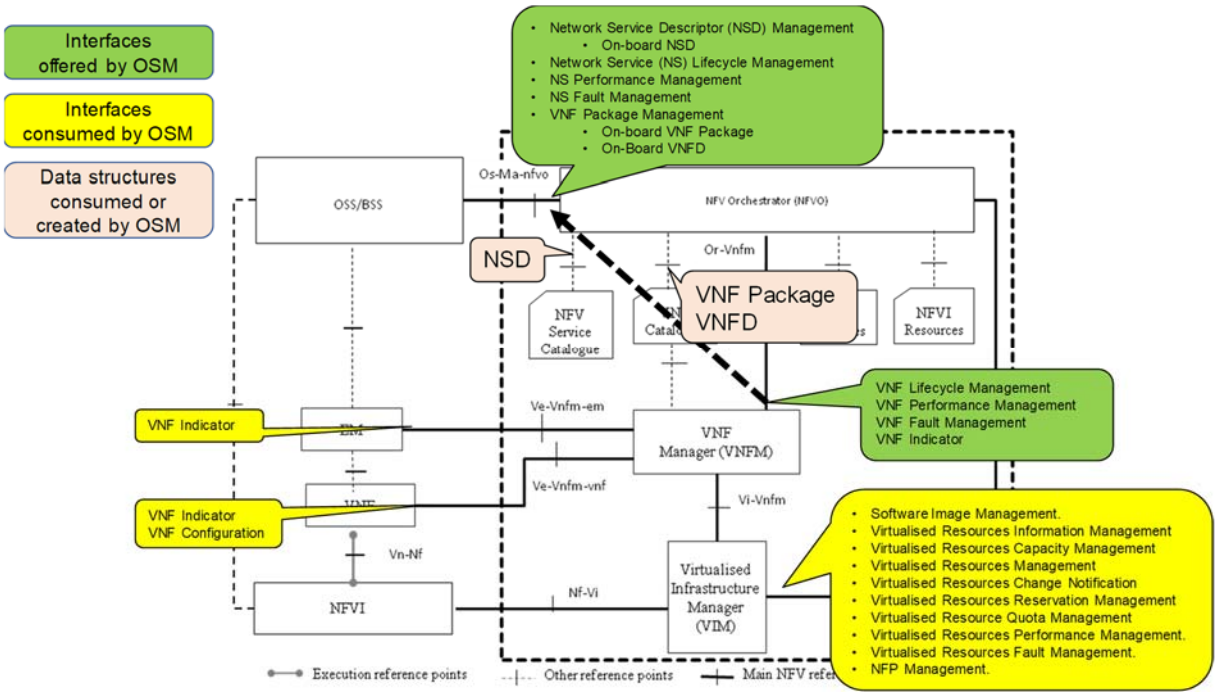

Figure 5 Interfaces and descriptors directly supported by OSM

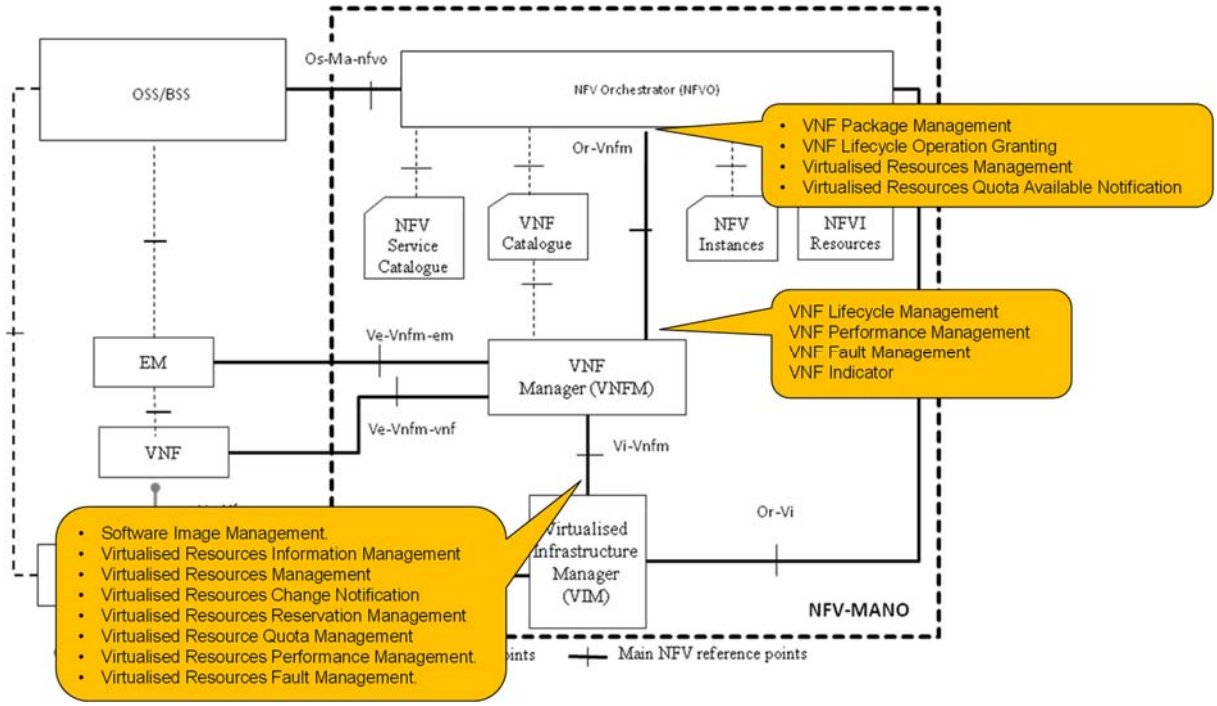

Figure 6 Interfaces not directly supported by OSM at this stage as a result of choices under IFA009.

We note that most of this interfacing which is not required by OSM has co-dependency on both sides of the reference point. For example, within the overall architecture, the VNF life cycle management interface offered by the VNFM is mutually dependent on the VNF package management, granting, and resource management interfaces offered by the NFVO. This co-dependency can be easily handled when the interfaces are internalised inside a single development, but if each side is offered by separate and independent developments, successful interworking relies on coordinated developments, which is extremely hard to obtain when solutions are coming from independent suppliers. As the very purpose of this interfacing is to allow separation of the development of the NFVO from the VNFM, this mutual dependency is likely to be a real and practical issue and one that is avoided by the OSM choices under IFA009.

3 Following ETSI NFV outputs is generally working well

There are many areas of the ETSI NFV output where OSM has been able to move to a successful implementation. This section aims to provide an overview of the salient specifications that have influenced OSM. This is not an exhaustive list of the material that has been created by the NFV ISG and material consumed by OSM, but serves to highlight the most relevant points.

The ETSI NFV Industry Specification Group and the ETSI OSM Open Source Group have fostered an open, transparent, supportive and productive collaboration environment. In this, OSM owes a special thanks to the leadership team in the NFV ISG (and the ETSI secretariat) who have appreciated the value that facilitating an open, bi-directional communication channel between implementers of the specification and those creating the specifications can bring to both parties.

3.1 ETSI NFV Phase 1 Specifications

ETSI NFV phase 1 specifications are officially deprecated and replaced by their phase 2 equivalents. However, there is still a lot of very useful directional material that OSM has leveraged from phase 1 to guide its journey.

Software Architecture (SWA) Working Group

The Software Architecture Working Group produced the *Virtual Network Functions Architecture specification* [Ref 1]. This specification has been formative in setting the direction for how VNFs are expected to be composed, what their most important lifecycle events are, and what the set of common states and state transitions would be needed.

Management and Orchestration (MANO) Working Group

The Management and Orchestration Working Group produced the *Management and Orchestration specification* [Ref 2]. This specification provided important key concepts for the role of MANO as a framework to support automated deployments of virtualized network functions. The primary reference points identified in this specification (while not all required in OSM) continue to be useful to convey where the functionality in the system resides. Most importantly for OSM, the VNF and Network Service Information Models identified in the specification were the starting point for the OSM data model and remain a relevant practical reference for some areas of the IM that were removed in the second phase.

Performance and Portability (PER) Expert Group

One of the outputs of the Performance and Portability Expert Group was the *Network Functions Virtualisation (NFV); NFV Performance & Portability Best Practises specification* [Ref 3]. This work was pivotal in establishing an understanding of the functionality that needed to be delivered by an NFV orchestrator that would be capable of enabling high performance NFV deployments. Some parts of the OSM data models were strongly influenced by the findings of this specification.

3.2 ETSI NFV Phase 2 Specifications

The ETSI NFV phase 2 specifications further refined and expanded on the concepts put forward for the industry in the phase 1 specifications.

Evolution and Ecosystem (EVE) Working Group

One of the outputs of the EVE working group is the *Network Functions Virtualisation (NFV); Virtualisation Technologies; Report on the application of Different Virtualisation Technologies in the NFV Framework. ETSI GS NFV-EVE 004* [Ref 7]. This has not yet impacted the delivered OSM codebase, but it is expected to become a useful reference in the upcoming releases as the community expands the capabilities of OSM to support container-based VNFs running on container based infrastructure technologies.

The Network Functions Virtualisation (NFV) Release 3; Licensing Management; Report on License Management for NFV, ETSI GR NFV-EVE 010 [Ref 8] addresses a really important topic to the OSM community. Although this topic has not yet been addressed in OSM, this report is expected to be used to guide the OSM community on its journey to integrate with next generation license management. Functionality such as license event reporting, and the necessity for the MANO stack to query a license management entity during key VNF lifecycle events, is likely to be implemented broadly aligned with this report recommendations once they have been fully processed by the Interfaces and Architecture (IFA) Working Group.

Interfaces and Architecture (IFA) Working Group

The Interfaces and Architecture (IFA) Working Group have both a considerable scope, and a considerable number of published specifications for the NFV ecosystem to absorb. Two of salient specifications are the *Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; VNF Packaging Specification, ETSI GS NFV-IFA 011* [Ref 11] and the *Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Network Service Templates Specification, ETSI GS NFV-IFA 014* [Ref 12]. IFA011 and IFA014 contain the definitions of the VNF Descriptor and Network Service Descriptor information models respectively.

Although section 4.2 notes some challenges that the OSM community has detected with adopting the specifications as-is, there is nonetheless a huge amount of value in these specifications as they go a long way to describing the consolidated needs of the VNF vendor and NFV operator communities. When the OSM community progresses the data model, it is with a very explicit view to keep the alignment with IFA.

Solutions (SOL) Working Group

The Solutions Working Group draft *Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point, Draft ETSI GS NFV-SOL 005* [Ref 10] is expected to have a very meaningful impact on OSM in Release FOUR. This document describes the RESTful API specification for the Os-Ma-Nfvo reference point. Although the OSM community considers there are some challenges with this specification (section 4.5), the OSM community is very much grateful for the initiative by the Solutions Working Group to articulate a normative definition of the northbound API, and hence pave a path for easier integration of NFV orchestration systems into OSS/BSS systems.

Testing, Experimentation and Open Source (TST) Working Group

The *Network Functions Virtualisation (NFV) Release 2; Testing; NFVI Compute and Network Metrics Specification, ETSI GS NFV-TST 008* [Ref 4] identifies a number of the metrics recommended for compute, network and storage NFVI subsystems. This specification has been very useful in informing the definition of the OSM monitoring module normalised NFVI metrics. Not all of the metrics identified in this specification are implemented in OSM at this time, as they are not available in current versions of the VIM level technologies that exist today, but still remain as a useful framework in case they these were eventually available (and therefore consumable by OSM).

The *Network Functions Virtualisation (NFV) Release 2; Testing; Guidelines on Interoperability Testing for MANO, ETSI GR NFV-TST 007* [Ref 5] provides an overview of a number of test cases, such as image, fault and performance management. These test case definitions have been used to inform the test cases that have been integrated into the OSM CI/CD pipeline that exercise a wide set of OSM functionality ranging from lifecycle management of images and VNFs, to the interfaces exposed by the OSM Monitoring module.

Reliability Availability and Assurance (REL) Working Group

The *Network Function Virtualisation (NFV); Reliability; Report on the resilience of NFV-MANO critical capabilities, ETSI GR NFV-REL 007* [Ref 6] provided an insightful view on the most important capabilities that OSM was required to deliver to be a service delivery platform with an appropriate level of resiliency. This work helped to solidify the OSM community approach to developing a theme related to service assurance where functionality such as supporting affinity/anti-affinity deployments was prioritised.

Security (SEC) Working Group

Securing the service delivery platform is of utmost importance to the OSM community. The output of the Security Working Group has been studied intensely. As an example, the *Network Functions Virtualisation (NFV); Trust; Report on Attestation Technologies and Practices for Secure Deployments, ETSI GR NFV-SEC 007* [Ref 9] describes characteristics of system integrity and capabilities related to securing the boot environment for the VNFs. Thus, there is an update in progress in OSM expected to land in an upcoming release that enables a measured launch environment once available from the VIM.

4 Observations on OSM Interfaces and Interoperability

4.1 Independent and Parsimonious Interfaces

Consistent with the choices under IFA009 as described under section 2 above, OSM looking south consumes APIs from VIMs, SDN controllers, and VNFs (either directly or through an EM). Looking north, OSM offers an API for on-boarding, control of life cycle, and in-life control of NSs and VNFs.

A strength of this approach is that all interfaces are APIs and there is no fundamental dependency in the development of each side of the interface. The API model means that the server side can define and produce its API and then at some later time, and without placing any direct constraints on the development and specification of the API, a client side can be developed to consume the API. **The client depends on the API but the API does not depend on the client**.

The other main objective for any interface is that it is parsimonious: the specification of the interface must include *only what is necessary* and *all that is sufficient*. There are distinct practical difficulties that arise when an interface specification is either unnecessary or insufficient and these may not be completely apparent until the interface is actually realised and used in practice. As an open source project, OSM is implementing the NFV interfaces (according to the OSM IFA choices) and is now learning directly about the necessity and sufficiency of the various specifications.

If a specification does not include all that is sufficient for its purpose, then the implementer needs to make additions sufficient to achieve proper operation of the overall system. However, each implementer is likely to make their own additions, which may well impede interoperability.

If a specification includes and requires more than is necessary, the user of the interface must have knowledge and complexity it does not need. For example, to instantiate and manage an NS, the client needs to know only the essential external features of the NS, the client does not need to understand the internal details of the VNFs and the topology of their internal interconnection. Conversely, if the interface requires the client to understand the internal details of the NS, the client is unlikely to be interoperable with NSs with equivalent external behaviour but different internal structure. For real interoperability, the interface specification must include only what is necessary.

Much of the rest of this white paper, especially as it relates to the OSM northbound interfaces, is effectively a discussion on OSM learning as to the *necessity* and *sufficiency* of the interface specifications for OSM purposes.

4.2 On-Boarding and Descriptors

The NFV descriptors, particularly NSDs and VNFDs are the blueprint that OSM uses to instantiate and manage the life-cycle of an NS or VNF. This means that NSDs and VNFDs should be *sufficiently* complete for OSM to carry out this process accurately and reliably. However, the descriptor should also not include parameters that create *unnecessary* restrictions on how the entity may be instantiated and managed. Overall, OSM makes two observations from its practical experience of working with NSDs and VNFDs:

- The specification of resource assignment for VNFs in the VNFD has changed significantly between the MAN001 and IFA011. While it has been argued that some of the specification relating to resource assignment in MAN001 was overly hardware specific and therefore

unnecessary which led to a greater level of abstraction in IFA011, it is clear to OSM that the level of abstraction in IFA011 means that some aspects of the IFA011 specification are insufficient for the accurate control of resources but IFA011 could be complemented by sections in MAN001 that were useful from the implementation perspective.

- At this stage, it is not realistic to have a fully automated on-boarding process which properly validates all actual end-to-end effects relying solely on descriptors developed based onIFA011/IFA014 specifications and outside the specific context of OSM. In practice, on-boarding requires a level of 'hands-on' work to test and refine a descriptor to ensure that it works as intended, and this requires a solid reference for testing the behavior as well as validating at scale that the behavior is consistent across different VIM or SDN technologies.

The latter point is discussed in more detail in section 4.3.

4.3 Descriptors and Controlling Resources and Performance

The hosting of VNFs has a number of distinct characteristics that differ from the hosting of standard cloud applications.

- The throughput performance of many VNFs will be bound by I/O and processing related to I/O operations, and the bottleneck may depend on careful allocation of hardware and specific hypervisor support. Explicit selection and control on these features is not normally needed in cloud applications and has not been generally available through standard VIM APIs.
- VNFs sit 'in the wire' rather than 'at the end of the wire', which tends to mean that more network interfaces are required on each VM, each connected to different networks. Thus, it is normally very important that the VNF correctly identifies each network interface (as created by the hypervisor) and that the interface is of the appropriate type (eg virtio, SR-IOV, passthrough, etc.) Cloud applications, however, tend to have a single network interface.

If it is indeed the case that the bottleneck for VNF capacity is different to many cloud applications, then there may be differences in the specific resource parameters that are important to controlling the throughput of the VNF, and these parameters may not be directly and fully configurable by standard cloud architectures. There is therefore a reasonable requirement for NFV to have the ability to configure and control these features in the NFVI.

However, there is a trade-off to be struck. A desire to configure and control specific features in the infrastructure can create dependencies and defeat the reusability of the NFVI, which is the object of the exercise. A VNF may gain an advantage in throughput if the server has specific hardware and the hypervisor has specific software drivers, or even if the VIM has specific configuration commands and fields to access these specific hardware and drivers. However, the degree of specificity that is allowed in the modelling of resources for a VNF needs to be carefully chosen to allow a reasonable degree of optimization without opening the door to over-specification of infrastructure requirements (e.g. requiring a very specific NIC model), which would defeat the benefit of the general-purpose infrastructure. If the cost of the saved resource from the improved throughput is less than the cost of the specialisation, the specialisation is simply not cost effective.

There is considerable cost saving when there is decoupling and independence on both sides. **VNFs are more valuable if they can be deployed on a wide variety of infrastructure** and therefore largely independent of the choice of infrastructure. It is separately and additionally the case that the **NFVI is more valuable if it can support a wide range of VNFs** and therefore independent of the choice of VNFs which are hosts. The real importance of this can be seen when it is clear that the NFVI will be deployed in advance of the selection of VNFs, and there the role of the NFV Orchestrator and its IM are key to facilitate that match.

In summary, there are some hardware, software, and VIM features that are worthwhile for supporting VNFs and there are even greater benefits if these features can be added as enhancements to the general-purpose cloud infrastructure (e.g. SR-IOV), but there are also many features which could enhance VNF performance, but are simply not worth supporting. Maximising VNF throughput performance is not an unconstrained objective and in a great many cases, the cost of an extra amount of server resource is likely to be much cheaper than specialisation and dependences in the layers of software sat above.

One of the features OSM has had all the way from the seed codebase was a lightweight but NFV specific VIM, OpenVIM. This has provided a useful reference for controlling the particular features of NFV infrastructure which are distinct from general cloud controllers such as OpenStack and understanding the overall cost optimisation between VNF throughput performance and total reusability of the NFVI infrastructure. However, the objective of OSM is not pursuing any kind of vertical integration, but that all those features can be gradually incorporated by OpenStack and any other appropriate VIMs, including VMware and even public cloud services[3].

The approach in OSM to the VNF performance problem is to identify a small set of parameters that, when incorporated into the VNFD, can be percolated from the NFVO to the VIMs, and then to the hypervisors, thus gaining an advantage in throughput and functionality without compromising the benefits of the general-purpose infrastructure.

Some of these EPA parameters are the following:

- Mempage size: size of memory pages to be allocated to the VM
- CPU pinning policy: whether the CPUs must be dedicated or can be shared
- CPU thread pinning policy: whether vCPUs must be mapped to physical cores, sibling HW threads or any HW threads
- NUMA awareness policy: whether memory, CPU and interfaces should come from the same NUMA node
- VM/VDU interfaces: SR-IOV, PCI-Passthrough, default VIM paravirtualized interface or emulated interface

OSM translates those parameters in the VNFD to specific parameters in the VIM flavors, particular to each VIM API. VIMs translate them to the appropriate hypervisor parameters that in the end guarantee that the VM is able to get maximum and predictable I/O performance.

[3] Although in this case, the trade-off with throughput performance may be even starker as public cloud services are unlikely to ever support the range of configuration available in a private cloud, even a standard one.

4.4 OSM Southbound Interfaces

In practice, **OSM has found that all the southbound interfaces it consumes effectively pre-exist and are not readily modifiable**. They are (with the possible exception of OpenVIM) under the control of large organisations with many often conflicting pressures (e.g. OpenStack, OpenDaylight, or vendors like VMware, Amazon Web Services, etc.), and even if there is a will to make specific technical changes to accommodate to specific requirements for that interface, many of these changes have conflicting consequences for the API. OSM has therefore taken the realistic approach of consuming VIM and SDN APIs 'as is'.

An immediate observation is that no VIM or SDN controller is specifically supporting the interface specifications of IFA005 or IFA006. Most, notably including the OpenStack API, support functionality that is broadly equivalent, but do not observe necessarily the NFV specifications. There have been several attempts to harmonise, or at least map, IFA005/IFA006 with the OpenStack API, so far with limited practical impact.

A further observation is that the APIs of the different releases of OpenStack are subject to changes and may differ from each other. Thus, the consuming clients are recommended to use the so-called "client libraries", maintained by the OpenStack project, so that the specific details of the messages sent over that interface become less relevant in practice, as software development processes is abstracted from those details, and on potential changes on that interface might have very limited impact for the client application. The fact that the client application can effectively rely on that piece of open source code is something that OSM has used extensively in its VIM plugins to interoperate more efficiently with a large set of OpenStack-derived VIMs and minimize development efforts over time.

In addition, it must be noted that there can also be slight differences among different vendor variations based on the same OpenStack release, with their own specific additions. Hence, even in those cases, a mere transposition of the client library is not sufficient, and a specific location in the code is required to support and normalize those slight (or large) variations among different variants and even VIMs of completely different nature.

All these have reinforced the approach followed in the OSM architecture, which can mediate to whatever API is presented —with the so-called "VIM plugin model" of OSM— rather than to insist on or wait for the suppliers of VIMs to conform to a single standard.

The same is true for VNFs. OSM has not yet encountered VNFs (or EMs) which are supporting directly IFA008 (or SOL002): **generally the VNFs which OSM has encountered have pre-existing management and configuration interfaces**. OSM has followed the same basic architecture of working to the actual APIs presented by the VNFs and has not insisted on or waited for interfaces directly following with the IFA standards. However, again the understanding from IFA007 and IFA008 has been very helpful and has been used to enable the internal abstraction of VNFs to enable the plug-in style of architecture, this time using "proxy charms"[4].

[4] Or Juju proxy charms.

4.5 OSM Northbound Interfaces

Looking northbound, with the API model, OSM does have control over the API specification and the requirement to interface with what currently exists does not give a constraint on OSM in the way it does with southbound interfaces. In principle, it would be possible for OSM to precisely conform to SOL005.

In OSM, the northbound API is the entry point to interact with the system and, therefore, new features in the system are always likely to require the addition of new API calls in that interface. OSM is committed to consider SOL005 as a preferential reference for NBI API calls in Release FOUR, understanding that some OSM functionality (particularly the one required to support its NS primitives) might require additional calls beyond its scope.

That said, such an exercise of alignment is not necessarily trivial even in some cases nominally covered in the spec. Whilst the syntax of the SOL specification is clear and precise, many of the operations themselves are very generic. An example is, under fault and performance management, the operation to 'subscribe to a monitor'. The details of what the monitor actually monitors and what parameters are used are not part of the specification. This means that the client subscribing to a monitor, in order to make any meaningful use of the monitor, must have prior knowledge of the monitor. Moreover, this cannot be discovered or otherwise semantically inferred as the information model of the interface offer no abstract semantic information about the monitor anywhere. In other words, the specification is insufficient.

OSM could offer such information on its API but this would involve OSM creating an additional semantic information model and offering an additional interface allowing clients to discover this information about the monitor. This is possible but, if done without contributing back to the ISG, would lead to a dilemma with neither choice being particularly satisfactory. If there is a general adoption of the OSM additions, then, de facto, OSM would become a parallel reference for industry in addition to the ETSI NFV ISG, which it does not seek to be. Furthermore, if other additions also emerged, the SOL specification might be no longer a reliable standard.[5] Observation about monitoring for fault and performance management is discussed in more detail in section 4.6.2.

We also note that the NS for in-life management (SOL005), in practice, is essentially a pass through to the constituent VNFs. This means the client using this interface needs to have a considerable amount of unnecessary knowledge of the detailed construction of the NS. This observation of the unnecessary aspect of the northbound interface is discussed in more detail in section 4.6 below.

In summary, we find we are still at the early stages of determining what will make for the ultimate northbound API for OSM and believe that this is case for NFV orchestrators in general. That said, it does not prevent OSM to take some design choices for its Release FOUR that begin this alignment

[5] It is useful to note that this is effectively what has happened with the OpenStack API when OSM looks southbound, where many vendors have added their own proprietary extensions and OSM has had to consider different cases into the same OpenStack plugin. However, in the case of OpenStack, the actual function being controlled and managed is common and understood – the different versions of OpenStack are all trying to access and control the same infrastructure. The semantics of what is being controlled is clearly understood by OSM.

with SOL005 while we point out what we understand as necessary refinements to make some aspects of SOL005 really useful in practice.

In order to minimize any potential ambiguity and facilitate as much as possible interoperability with a given OSM release**, OSM has decided to continuously expose in OpenAPI format the models that conform to the accurate description of its northbound interface**. In addition to the obvious advantages in terms of transparency (e.g. discussions on NBI progression per release can be effectively driven at this repository), the developer of client applications can benefit from the capabilities that Swagger/OpenAPI offers to auto-generate client code for their application in - literally - almost any potential programming language of choice. Hence, there is no room for ambiguity neither for general users or OSM developers, since whichever that is described in that repo is OSM's NBI and vice-versa.

4.6 Presenting and Managing an NS as a Single Entity

When a VNFM presents through its northbound interface in accordance with IFA007 control of a VNF, it is presented as a single VNF, even if the VNF itself is made up from many VNFCs/VDUs and VLs. Specifically, the combination of the VNF/EM and the VNFM act to coordinate control messaging for the individual components of the in-life VNF such that, northbound, the VNF appears as a single coherent entity. Broadly there are four areas of control of the VNF as illustrated in Figure 7, namely:

- Configuration – including initialisation of the VNF when instantiated ('day 1') as well as changes to configuration in-life ('day 2');
- Resource management – to control the capacity of the VNF;
- Fault management;
- Performance management.

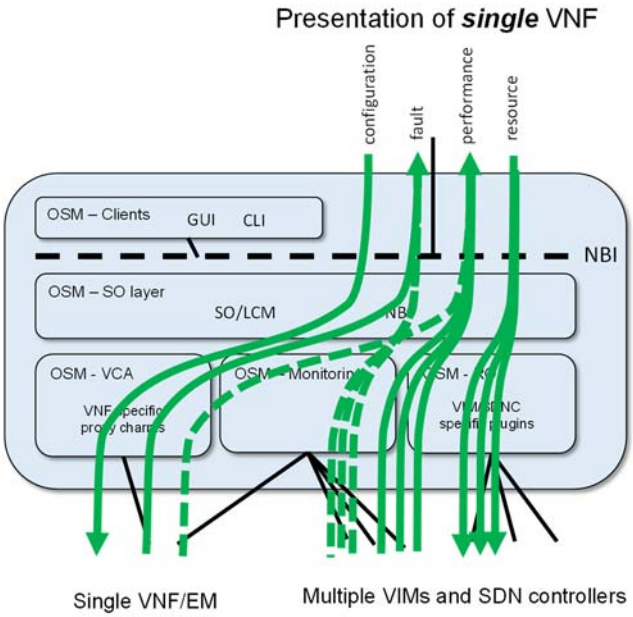

Figure 7 Presentation of the control of a VNF as a single interface

In contrast, in accordance with IFA013/SOL005, the northbound interface from the SO layer to control an NS presents the in-life NS as its constituent VNFs and does not present a single coordinated control of the NS: the northbound interface for the control of the in-life NS is a simple pass-through to the component VNFs and VLs of the in-life NS as illustrated in Figure 8.

There appears to be no meaningful interface for any of the areas of in-life control of the NS as a single NS (rather than a collection of independent VNFs and VLs) nor for the specification and control of the capacity of the NS as a single NS (again rather than as a collection of independent VNFs and VLs)[6].

[6] IFA013/SOL005 might appear to allow the possibility of some coordination of the fault management and performance management as the monitors are untyped and have arbitrary identifiers, IFA013 states "An alarm on a given NS results from either a collected virtualized resource fault impacting the connectivity of the NS instance or a VNF alarm, resulting from a virtualized resource alarm, issued by the VNFM for a VNF that is part of this NS instance." A reasonably interpretation of this if that the NS fault reporting is simple a pass through fault reporting from the constituent VNFs, VMs, and VLs.

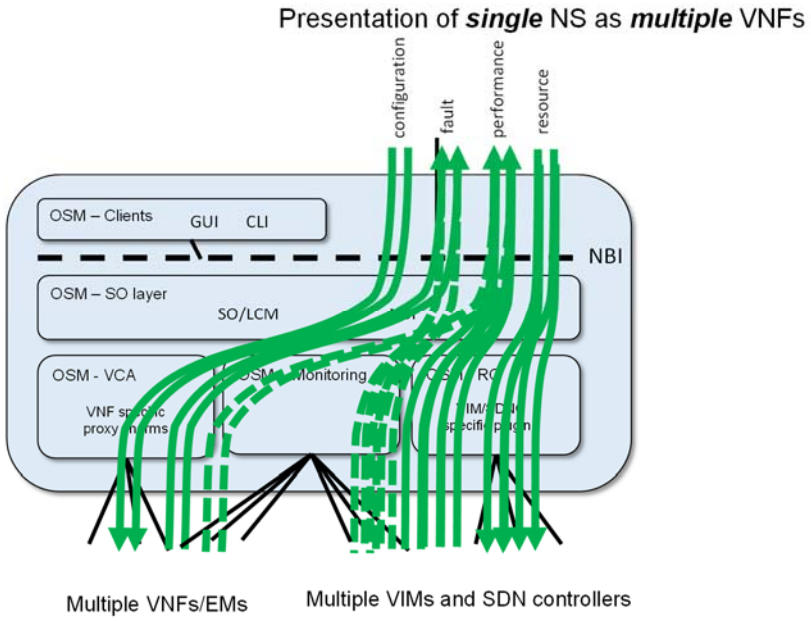

Figure 8 Presentation of the control of an NS as a pass-through the control of the constituent VNFs and VLs

This solution is not ideal for these important reasons:

- it does not give expression to the holistic properties of an NS that are beyond the properties of the components and that can occur in all four of the areas of control – configuration, capacity, fault, and performance;
- the control of the NS is not being properly and fully automated as the rules and policies by which the NS can be modified are not present in the NFVO system. They are held in some arbitrary and unspecified system above the NFVO;
- changes to the in-life NS that do not affect the external behaviour of the NS (for example resilient topology) are not encapsulated and are fully exposed;
- it does not scale as the number of entities being controlled, and it will rise multiplicatively as layers of orchestration are added (indeed, encapsulation and presenting the composite as a single entity is the essence of scalability in terms of management);
- it is not recursive as the orchestrator at each layer must be specific to that layer as it must understand how many layers there are to the constituent VNFs.

An effective solution would have the northbound control interface for an NS as presenting the NS as a single entity and would essentially be the same as the northbound control interface for the VNF as illustrated in Figure 9.

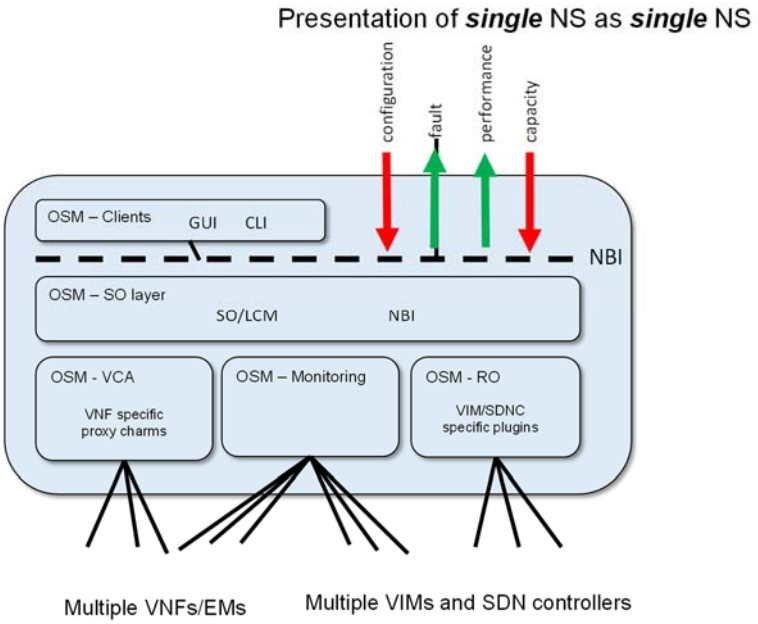

Figure 9 Presentation of the control of a NS as a single interface

4.6.1 Configuration and Capacity Processes

In order for the control interface of the NS to be parsimonious, that is necessary and sufficient, an agent that can translate parameters is needed. The agent will reflect appropriate external configuration of the NS as a single holistic entity into all the internal configuration changes needed within the NS. Based on what is implied by the IFA013/SOL005 interface, this agent is not currently assumed to be part of the NFVO or part of the NS: the implication is that it sits outside and above the IFA013/SOL005 reference.

As noted above, this neither solves the problem - it is just pushed elsewhere - nor does it scale, as encapsulation - the essence of scaling - never takes place.

A particular example of this is making allowed modifications to a network service in-life. For example, suppose an NS instance currently comprises of a router VNF and a firewall VNF and a modification is requested to add intrusion detection VNF to this NS instance. Rebuilding a new NS instance from scratch is not an ideal solution as this may involve both a substantial service outage and losing the detailed firewall policy.

On the other hand, if there is an NS configuration agent, this can validate the change request, calculate the minimum difference between the current NS and the desired NS, and have an effect on the minimum changes.

In addition, the parameterisation and control of the basic capacity of an NS is entirely absent. The effective throughput of NS is an important characteristic of the NS but cannot be considered to be

some sort of arbitrary aggregate of the capacity of the VNFs[7]. The actual throughput of an NS depends on the topology of the NS, the different forms of traffic which the NS is asked to handle and the forwarding paths for each form of traffic through the topology. Even when topology is properly accounted for, one needs to consider possible scaling of any one VNF either by changing the topology of the VNF or by adding resources to individual VNFCs within the VNF.

The implication is that the specific calculation of the capacity of the NS is specific to each NS and cannot be simply dismissed or assumed to be some default conversion factor. Coordinating control functionality is unavoidable and should be directly associated with the control of the NS as illustrated and summarised in Figure 10.

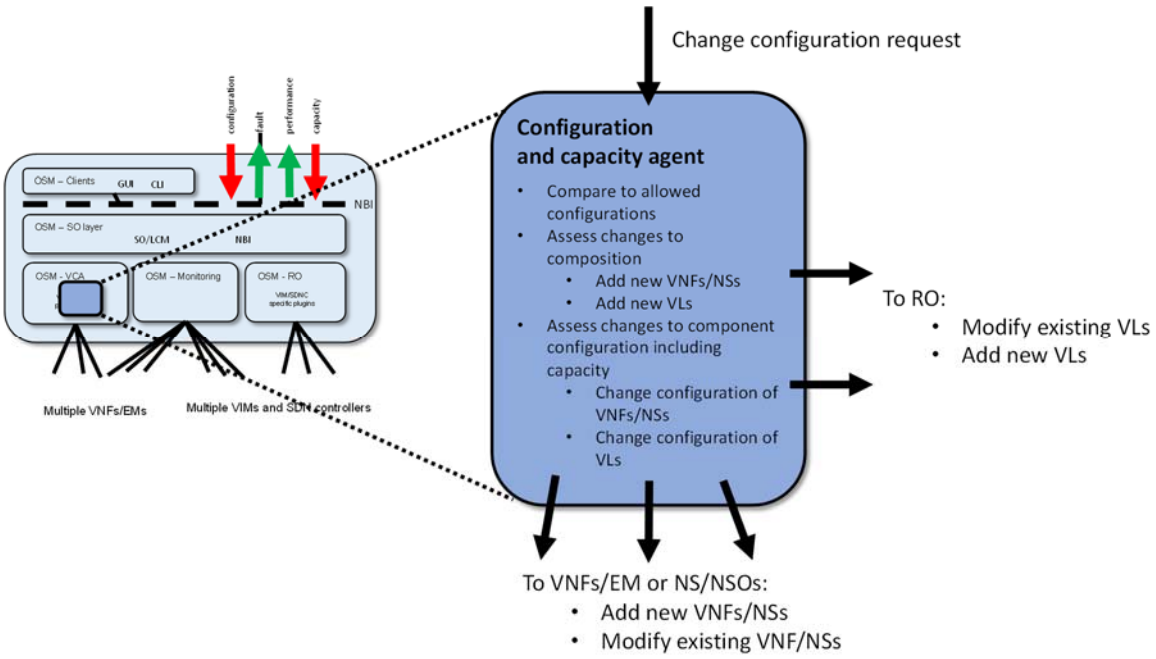

Figure 10 NS configuration and capacity management agent

One possible solution might be to create a full VNF that performs this coordinating function and this VNF would present the coherent single configuration and capacity management interface. This would be broadly consistent with the current interface specifications, however, there would need to be some mechanism for restricting the access to the configuration of the individual VNFs and ensure that all configuration is carried out via the configuration VNF. However, there is much of the framework for this functionality that is common to any NS and this solution would not encourage such a common framework. In addition, the development of the VNF configuration may be a huge

[7] In fact, in the same way, the capacity of any VNF cannot be considered to some sort of arbitrary aggregation of the VMs supporting the VNFCs of the VNF. Parameterisation of VNF capacity is also a significant gap in the current IFA007/SOL003 interface standards.

challenge in case of dealing with a NS that involves the usage of VNFs coming from different vendors, which is a likely and desirable scenario for most service providers.

The other possibility is to have a generic NS manager, analogous to the generic VNFM, and which could be implemented in a similar way to the generic VNFM.

4.6.2 Fault and Performance Management Processes

In a very similar way, instead of presenting the fault and performance information of each VNF separately, an NS coordinating function is needed to bring together information from the constituent VNF, understand their impact in the context of the NS topology, and report the health of the NS in terms meaningful to the NS. This is illustrated in Figure 11 below.

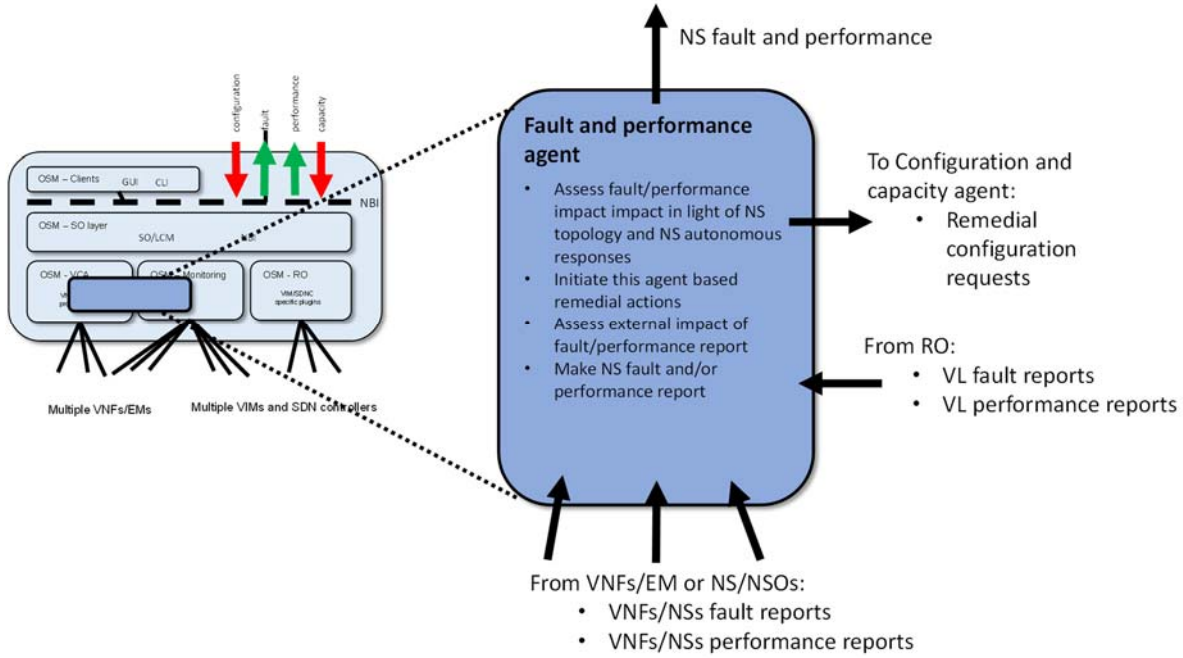

Figure 11 NS fault and performance reporting agent

A further aspect of fault and performance management that is highlighted by this is that the information is used for different purposes and each user needs a different perspective on the information. There are at least three major users of the fault and performance information.[8]

[8] A simple and possibly more familiar illustration of these three separate processes is with a RAID array of disks. If a disk fails:

- the RAID system will use its redundancy to maintain operation (the automated restoration agent);
- a maintenance alarm is raised to replace the failed disk (the repair process);
- the RAID system may report 'degraded' operation to its users to alert them that the RAID array is running with reduced resilience (service status reporting to users).

- **Automated restoration agent**. The objective of this agent is to maintain the functionality and capacity of the NS (or whatever is the functional service in question, for example a VNF or a VL). This process is not primarily concerned with understanding the cause of any fault or degradation, only in making good by using whatever resources it has available.
- **Repair process**. This is the process that is concerned with understanding root causes and repairing those resources that have failed. Assuming there is a separate automated restoration process, the service status should not be directly affected by the speed of the repair process.[9]
- **Service status reporting to users**. The process reports the status of the function as it appears externally and impacts the user.

4.6.3 Recursion as an Alternative to Granting Between the NFVO and the VNFM

Finally, as well as addressing the issues on necessity and sufficiency of the OSM northbound for NSs, presenting the control of NS as a single NS rather than the collection of VNFs and VLs would also give an alternative solution to the long-standing ambiguity of the split of logical functionality between the VNFM and the NFVO.

With this solution, even the so-called "Specific VNFMs" could be treated as if they were an NFVO, and there would be no need for the complex system of granting. The S-VNFM, acting as a full NFVO, could have full control of the resource allocation without constantly deferring for permission.

However, as it has been described above and shown in Figure 12 below, this requires that the NS northbound control needs new interfaces for NS configuration and NS capacity management. Likewise, the incorporation of some aspects from SOL003 in OSM's NBI is considered as a path worth exploring in order to ease this symmetry, and will require further study in coming OSM releases.

[9] A slow repair process may, for example, put the function are greater risk of multiple simultaneous failures that will cause the automated restoration process to fail.

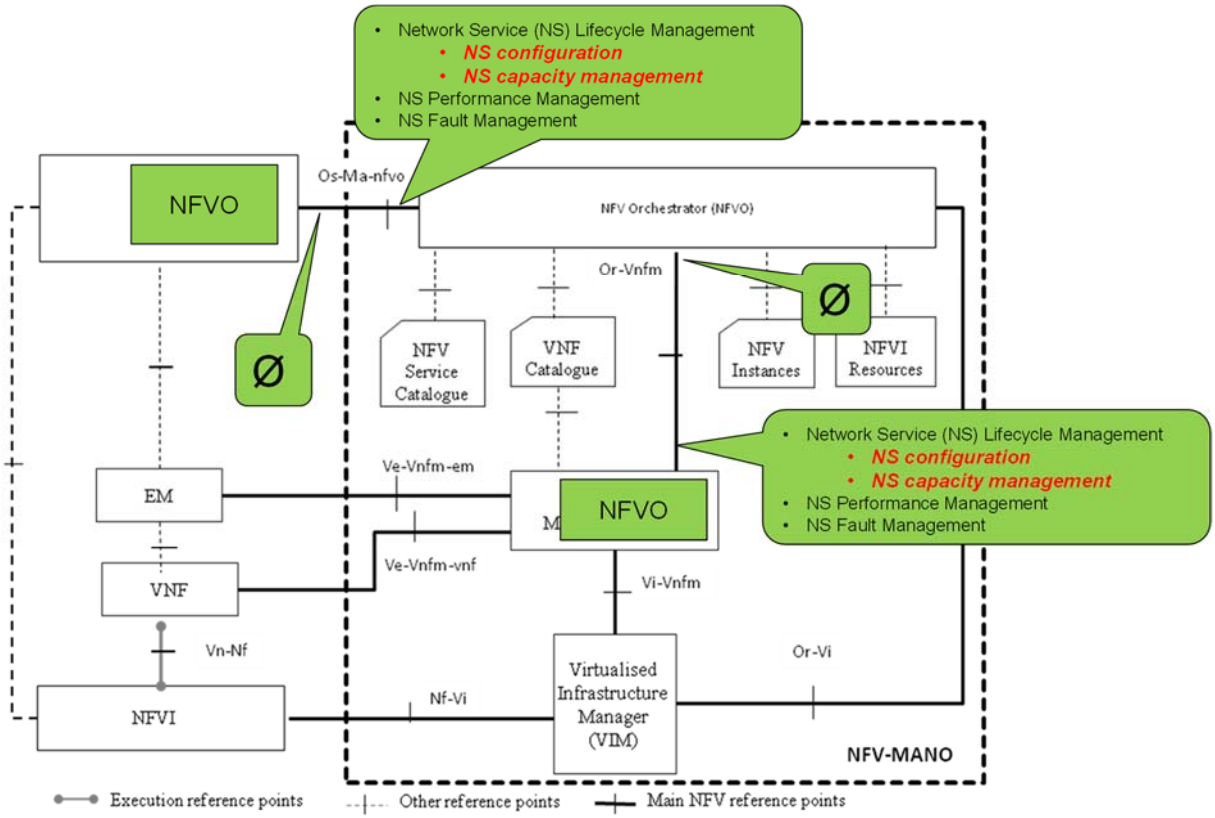

Figure 12 Recursive interfacing for the control of NS with additions required (in red)

5 Challenges for the Carrier, Vendor, and Academic Communities

5.1 Alternative Representations of the ETSI NFV Architecture

The NFV architecture diagram first emerged in the first round of ETSI NFV documents in October 2013. At the time, it was the subject of much discussion and was, even at the time, something of a compromise, in particular accommodating differing views expressed by the SWA and MANO working groups. Having said that, it has endured well, is used frequently, and instantly recognised by most of the industry involved in NFV.

However, this diagram was never envisaged as the last word on NFV architecture and what we have learnt in OSM is that there are a number of important architectural aspects that require more detail than is present in this diagram. As a result, we believe that it is important to augment this diagram with more details taking into account a number of different perspectives on the architecture.

- ***Different perspectives for logical functionality and implementation domains.*** It would be helpful to develop two separate perspectives within the architecture. The first perspective would describe only the aspects of the architecture that are about the logical functionality whilst the second would describe only aspects of the domains of responsibility for implementation of the logical functionality. This would eliminate much of the current

ambiguity in the current architecture between the EM, the VNFM, and the NFVO. Whilst the current architecture describes these as logical functions, in practice, their scope is actually defined by a domain of implementation responsibility. For example, the EM is within the implementation scope of the VNF developer (essentially by definition) whilst the logical functionality of any EM is left open and different VNFs have widely different logical functionality within their EMs. Whilst many VNF implementations place the functionality that supports fault and performance reporting in an EM, some VNF implementations choose not to have an EM and have this functionality within the VNF itself. The example of the logical functionality associated with the scaling of a VNF according to required capacity load is especially problematic. This functionality could reasonably be implemented in the VNF itself, an EM, a VNF specific VNFM, a generic VNFM, or the NFVO. This ambiguity in where logical functionality is implemented in turn means that the meaning of messages on particular reference points, when viewed in the context of the overall automation process is also ambiguous. Moreover, this in turn means that it is very difficult to design an overall automation process that accounts for all the possible allowable choices made by implementers on where to place logical functionality.

- ***Inclusion of consuming 'as a service' within the architecture.*** The concept of 'as a service' is very well established in cloud and has been a major part of the success of cloud architecture. This has been incorporated to some extent in the NFV architecture, however, this has mainly focused on NFV offering something 'as a service', for example NSaaS, or VNFaaS, or even NFVIaaS. However, the architecture is weaker on how an NFV environment can consume something 'as a service'. An essential property of 'as a service' is that the functionality that delivers the service is largely encapsulated and its details are hidden from the client of the service. However, this encapsulation property has appeared to run counter to aim of the NFVO to be able to fully optimise the use of resources and a view that in order to achieve this optimisation, the NFVO must be able to see and control the detailed resource allocation. This latter view has tended to hold sway in discussion on this matter. However, even if it is generally true (and there is good reasons to believe that it is not – there is a danger of spending a Euro or Dollar to save a Cent), it does not alter the reality of carrier's environments which make consuming 'as a service' a practical necessity for evolutionary interworking and inter-carrier services.
- ***Description of end-to-end automation processes.*** It would be helpful to develop documentation of the overall automation processes based only on the logical functionality. If NFV is to achieve its primary aim of automating the instantiation of complex services, the absence of any documentation of these processes is a significant obstacle to achieving this aim, especially where interworking is required.

There is no need to change the existing architecture to address these aspects. However, the absence of architectural clarity on these points is a significant obstacle for carriers and other end users in deploying and exploiting NFV and achieving the real potential of NFV for process automation.

5.2 NFV Requirements Beyond Cloud Implementations

There are subtle but important differences between cloud applications and NFV VNFs and NSs, which create challenges if existing cloud automations systems are to be exploited directly by NFV.

In particular, most cloud applications are 'end applications' whilst most VNFs and NSs sit 'in the wire'. Whilst most cloud applications need only one network interface connected to one network[10]. In contrast, almost by definition, a VNF or NS will sit 'in the wire' and will sit as a gateway between two or more networks and is likely to have different interfaces attached to these different networks.

In addition, the critical bottleneck resources that control the capacity of a cloud application and a VNF are quite likely to be different, for example, a cloud application typically is limited by processor capacity or memory while a VNF normally is limited by I/O capacity. This means that pragmatic resource abstraction for cloud may not be ideal for VNFs and NSs. Furthermore, there is a challenge in balancing the objectives of direct reuse of cloud automation systems (including now those that support containers, such as Kubernetes) and efficient support of NFV, which are likely to require an appropriate and non-trivial IM progression in order to enable a harmonious combination of the best of both technologies.

5.3 Conformance Testing

The use of API frameworks such as REST and YANG with some of their associated support tools (for example, Swagger) make possible the automation of conformance testing of an implementation of an API specification. This is helpful and can eliminate many basic errors in API implementation and as such is very welcome and a significant benefit to the implementation of APIs.

However, it is equally important to remember that this conformance testing is only verifying the lower layers of the protocol. This creates a double danger. First, the higher layers of the protocol that interact with the system wide semantic aspects of the interface require separate testing which is not automated in the same way and may be simply ignored. Second, there is a danger that users of a system offering an interface with verified conformance mistakenly believe that this conformance implies and guarantees interoperability. There is a danger that people assume conformance is the same as interoperability, and can even use this confusion deliberately to claim the blind adherence to an interface spec without really providing the E2E functionality that is expected.

This latter danger could be significantly mitigated if the NFV architecture is augmented with descriptions of the end-to-end automation processes as discussed above. However, if they are absent from the architecture and absent from the conformance testing, this presents a considerable challenge for achieving proper interoperability.

5.4 Ownership of Standards and Specifications and evolution of information models

Within NFV there are many essential interfaces and many protocol layers to each interface. When any change or addition is needed to an interface specification there is a real challenge in deciding which body should be invited to make appropriate amendments to a particular specification or if several groups and SDOs are required to operate in a pipeline of decisions (that must be agreed in a cascade of formal agreements in several consensus-driven organizations).

[10] And even if more than one network interface is used to support large capacity applications, these are still normally seen as a single logical interface either by using load sharing or a subnet address rather than a single end address.

This issue arises notably when a specification is based on layers of templates and, particularly, in the progression of the MANO information model, including the descriptors, that are key to feed the NFVO system. Thus, while the NSD and VNFD are semantically defined (as information model) in ETSI NFV's IFA011 and IFA014, they are expected to be syntactically defined (as data model) in ETSI NFV's SOL001, which, in turn, should be based on some specific templates from OASIS TOSCA (derived from more generic templates developed for cloud services rather than NFV services).

Being this pipeline of decisions sufficiently complicated to evolve the models, things are even more challenging in practice with TOSCA, as it happens that those OASIS templates assume implicitly a different information model, contained in the so-called "TOSCA Simple Profile for NFV", which is maintained inside OASIS. This latter TOSCA Simple Profile for NFV is a data model and is not necessarily bound to any evolution of the IM of IFA011/IFA014, leading to the effective discontinuity of an already complex pipeline of decisions, and, in the best case, should involve detailed liaison covering the technical subtleties across many working groups. This is a challenge for creating specifications in a timely manner, and an even greater challenge if they need to evolve with the pace of new software technologies especially in the field of virtualization, where continuous innovations are introduced every year. **This approach is leading to the growing proliferation of proprietary changes and extensions by vendors in order to make their systems at least work with their own products**. It is understood that, from the carrier and end users point of view, this introduces another layer of ownership of the specification – the implementing vendor— that would hamper any reasonable expectations of effective interoperability.

On the other hand, there is a second adverse effect - briefly discussed before - which is augmented by such a complex pipeline: **the error of identifying conformance with interoperability and, furthermore, using an ambiguous/partial reference of conformance to suggest interoperability**. In a pipeline like the one described above, there are many stages which correspond to something that can be described as TOSCA, specifically:

- Any IM described using OASIS TOSCA as modelling language
- The IM implicit in those generic templates developed for cloud services or a variant of them
- The IM implicit in "TOSCA Simple profile for NFV"
- The IM changes being pushed by SOL005 and not adopted in TOSCA Simple (or a fork of them maintained by SOL005?)
- Any interpretation of IFA IM directly written in TOSCA
- Any vendor-specific IM, with proprietary changes and/or extensions, that is perfectly conformant to TOSCA modelling remembering that, in this context, it is legitimate to over exploit object inheritance and to define a "Vendor X Virtual Machine" type, "Vendor X Connection Point" type, "Vendor X VNF" type, etc., which is **a path which may prevent effective interoperability at IM level**.

The challenge is that this confusion might well lead to a loss of interoperability, a statement can be made which claims conformance giving the misleading impression of interoperability. Thus, all the cases above could legitimately claim compliance to something tagged "TOSCA" even if there are different and incompatible IMs implicitly assumed. This is also true when looking at implementing an IM; simply stating conformance to "TOSCA" is insufficient to specify a target DM for an IM.

In this confusing landscape for the development of the IM and DM, OSM has decided to simplify things for the end users and bring as much clarity as possible, avoiding any source of ambiguity from the perspective of interoperability. At the same time OSM keeps pushing for harmonization among the different groups described above. In that respect, OSM has opted for a transparent approach here:

- **Clarity on the Information Model is the key to guarantee genuine interoperability of implemented interfaces with actual DMs especially given the pace that different features and technologies need to be enabled and supported by OSM**, regardless the modelling language of choice to write down that IM (which, at the end of the day, could be automatically translated while preserving the IM underneath).
- **OSM has decided to continuously expose its information model in a special repository from which code is auto-generated** and is imported by its different modules. This has numerous advantages in terms of transparency:
    - Discussions on IM progression per release can be effectively driven at this repository, where comments, suggestions, etc. can take place openly.
    - An auto-generated and documented HTML map of the IM is auto-generated, and always available at OSM's website.
    - There is no room for ambiguity for either general users or OSM developers. Whichever that is described in that repo is OSM's IM and vice-versa.
- OSM wants to avoid any further contribution to the confusion around the "different TOSCAs" described above, and has deliberately chosen to describe its IM in another widely supported modelling language, YANG, which does not bring any implicit assumption around the IM. Thus, OSM avoids claiming conformance when in fact interoperability is not clearly established and focuses on enhancing the standardized IM/DM as the key for actual interoperability.
- At the same time, OSM regards ETSI NFV ISG as a central point to facilitate convergence in this debate, and is committed to keep sharing its findings with the ISG community.

5.5 Engagement of Carriers and Other End Users

Open source projects provide a new opportunity for standards bodies: they provide an open environment for testing and validating standards, a step which is most often currently carried out within the closed and proprietary domain of each vendor.

In particular, this open environment of the open source project gives the opportunity to test directly for the completeness of the specifications and form a view on both ambiguities in the specifications or any over specification. Indeed, it is well known that there is a risk that the commercial interests of vendors may be served by ambiguous or incomplete specifications, especially if two or more vendors already have implemented slightly different solutions and if a little ambiguity or incompleteness would allow both to assert compliance to the specification. However, interoperability may be compromised and this might not become apparent until sometime later. At that point, a carrier may be faced with either developing an expensive interworking solution or purchasing components from only one vendor whose components are interoperable with each other.

Open source offers carriers and other end users the possibility to understand and test the completeness of a specification and learn where points of ambiguity may arise as well as spotting aspects of specification that are unnecessarily restrictive. The open source project is a place where carriers and end users can work directly with vendors and others, working on implementation including academics and research projects to develop clear and robust interoperability. However, this requires a level of active engagement from all the players, and especially from carriers and other end users.

Ideally, this involvement should extend beyond setting and/or endorsing requirements and even if this does not involve actively producing and committing code for the project, this should involve at least a level of active engagement with the development of a more detailed system architecture, as well as testing the system as it is being developed, in order to offer informed and constructive feedback.

Involvement by carriers and other end users in this way can give constructive feedback at a stage in the development process at which changes can be made with little cost or impact on the system development.

6 Annex: Use Cases for Presenting the NS as a Single Entity

6.1 Uses Case: Orchestration of Orchestration

There are a number of practical situations that can arise when one orchestrator cannot orchestrate an entire service end-to-end, as often the delivery of a service to a customer will need the involvement of more than one orchestrated domain. This can happen within a single network operator's network and certainly would exist if an end-to-end service required components from another network operator.

For example, a network operator may have major platforms each capable of constructing network services and each platform supplied by different vendors and supplied with its own orchestrator. The network might then wish to construct an end-to-end service within an end to end orchestrator. This leads to an architecture of orchestration of orchestrators.

An example between operators might include an end customer who wants a service in multiple geographical locations, and where no operator has physical presence in them all. Another example might be that some operators specialise in end-customer-facing operations, whilst other operators specialise in the "wholesale" provision of infrastructure, or in providing specific types of VNF.

This suggests cooperating orchestrated platforms should be organised in a hierarchical architecture, meaning that an upper (or client) platform provides the end-to-end service to the customer, and it chooses to involve a lower (or server) platform to deliver part of the required capability. That is, they have a north-south and not an east-west relationship. From the customer's perspective, they only interact with, and know about, the upper platform; from the upper platform's perspective, the lower platform is providing a component in their overall network service in a similar manner to the NFVI. As far as the lower platform is concerned, the upper platform is just another customer requesting service. Ideally, this architecture should be fully recursive, meaning that the lower platform can in turn arrange for some of the service it provides to be delivered by a yet lower platform, and so on as needed. Alternatively, the upper platform may find it is supplying a service that is a component of some even higher platform.

The advantages of such an approach are commercial and technical: it has clear lines of responsibility, allows flexibility in service provision and vendor supply, and only a single, standardized north-south API is needed.

6.2 Uses Case: Automated Modification of Firewall and Load Balancer Rules

Consider the relative simple example of an NS that is capable of supporting several service chains and these service chains are accessed via a firewall VNF. This firewall VNF controls the access to the different service chains according to firewall policies. In principle, the network service can be modified in-life to support new service chains modify service chains or delete service chains and when this happens, the firewall policies need to be changed.

The modification of the firewall policies is a direct consequence of the change to the service chains and so, externally, there is no necessity to expose the firewall policy configuration. Externally, the

requested of the network services only need to issue the request, for example, for a new service chain, and the network service should have all the necessary information to adjust the firewall policy accordingly.

This is illustrated in Figure 13.

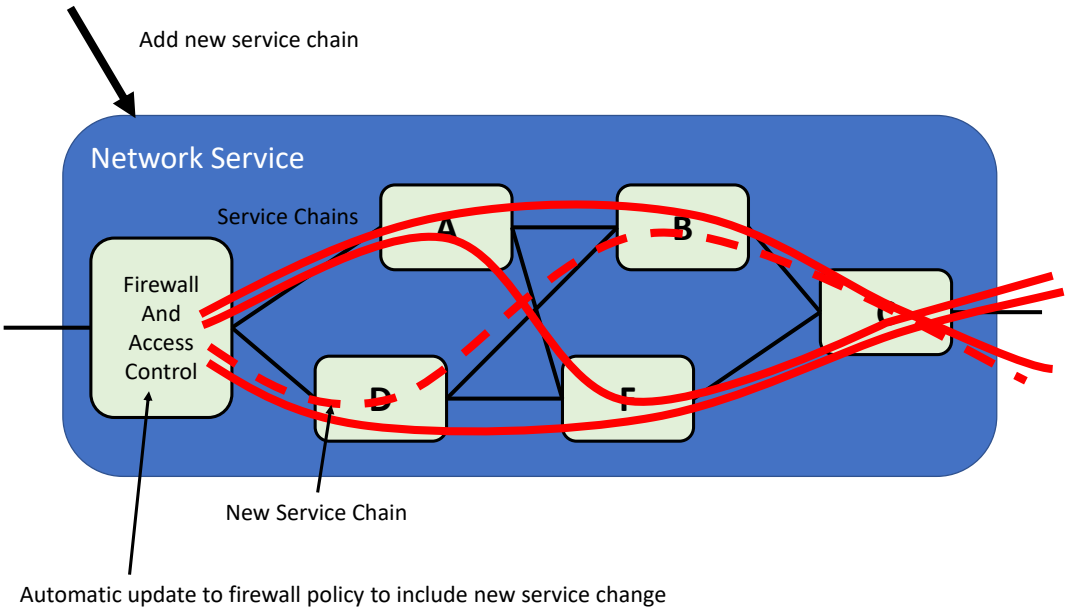

Figure 13 Consequential adjustment of firewall policy following addition of new service chain.

However, the current NFV architecture does not include any clear obvious place where this logic can be housed within the NS. Indeed, the way the interface standards imply simple pass through would suggest that such functionality would sit outside the NS and would have to be part of the logic provided by the user of the NS.

This particular use case gives a good illustration of the absence of proper encapsulation of the NS leading to configuration being unnecessarily exposed externally.

7 References

1. Network Functions Virtualisation (NFV); Virtual Network Functions Architecture. ETSI GS NFV-SWA 001 V1.1.1 (2014-12), http://www.etsi.org/deliver/etsi_gs/NFV-SWA/001_099/001/01.01.01_60/gs_NFV-SWA001v010101p.pdf
2. Network Functions Virtualisation (NFV); Management and Orchestration. ETSI GS NFV-MAN 001 V1.1.1 (2014-12). http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf
3. Network Functions Virtualisation (NFV); NFV Performance & Portability Best Practises. ETSI GS NFV-PER 001 V1.1.2 (2014-12). http://www.etsi.org/deliver/etsi_gs/NFV-PER/001_099/002/01.01.02_60/gs_NFV-PER002v010102p.pdf
4. Network Functions Virtualisation (NFV) Release 2; Testing; NFVI Compute and Network Metrics Specification. ETSI GS NFV-TST 008 V2.1.1 (2017-05). http://www.etsi.org/deliver/etsi_gs/NFV-TST/001_099/008/02.01.01_60/gs_NFV-TST008v020101p.pdf
5. Network Functions Virtualisation (NFV) Release 2; Testing; Guidelines on Interoperability Testing for MANO, ETSI GR NFV-TST 007 V1.1.1 (2017-11). http://www.etsi.org/deliver/etsi_gr/NFV-TST/001_099/007/01.01.01_60/gr_NFV-TST007v010101p.pdf
6. Network Functions Virtualisation (NFV); Reliability; Report on the resilience of NFV-MANO critical capabilities, ETSI GR NFV-REL 007 V1.1.2 (2017-10). http://www.etsi.org/deliver/etsi_gr/NFV-REL/001_099/007/01.01.02_60/gr_NFV-REL007v010102p.pdf
7. Network Functions Virtualisation (NFV); Virtualisation Technologies; Report on the application of Different Virtualisation Technologies in the NFV Framework. ETSI GS NFV-EVE 004 V1.1.1 (2016-03). http://www.etsi.org/deliver/etsi_gs/NFV-EVE/001_099/004/01.01.01_60/gs_NFV-EVE004v010101p.pdf
8. Network Functions Virtualisation (NFV) Release 3; Licensing Management; Report on License Management for NFVETSI GR NFV-EVE 010 V3.1.1 (2017-12). http://www.etsi.org/deliver/etsi_gr/NFV-EVE/001_099/010/03.01.01_60/gr_NFV-EVE010v030101p.pdf
9. Network Functions Virtualisation (NFV); Trust; Report on Attestation Technologies and Practices for Secure Deployments, ETSI GR NFV-SEC 007 V1.1.1 (2017-10). http://www.etsi.org/deliver/etsi_gr/NFV-SEC/001_099/007/01.01.01_60/gr_NFV-SEC007v010101p.pdf
10. Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point. Draft ETSI GS NFV-SOL 005 V0.12.0 (2017-12). http://docbox.etsi.org/ISG/NFV/Open/Drafts/SOL005_Os-Ma-nfvo_APIs/NFV-SOL005v0120.zip
11. Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; VNF Packaging Specification, ETSI GS NFV-IFA 011 V2.3.1 (2017-08). http://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/011/02.03.01_60/gs_NFV-IFA011v020301p.pdf

12. Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Network Service Templates Specification, ETSI GS NFV-IFA 014 V2.3.1 (2017-08). http://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/014/02.03.01_60/gs_NFV-IFA014v020301p.pdf

ETSI (European Telecommunications Standards Institute)
06921 Sophia Antipolis CEDEX, France
Tel +33 4 92 94 42 00
info@etsi.org
www.etsi.org

This White Paper is issued for information only. It does not constitute an official or agreed position of ETSI, nor of its Members. The views expressed are entirely those of the author(s).

ETSI declines all responsibility for any errors and any loss or damage resulting from use of the contents of this White Paper.

ETSI also declines responsibility for any infringement of any third party's Intellectual Property Rights (IPR), but will be pleased to acknowledge any IPR and correct any infringement of which it is advised.