ATIS-I-0000044

Operational Opportunities and Challenges of SDN/NFV Programmable Infrastructure

October 2013

As a leading technology and solutions development organization, ATIS brings together the top global ICT companies to advance the industry's most-pressing business priorities. Through ATIS committees and forums, nearly 200 companies address cloud services, device solutions, emergency services, M2M communications, cyber security, ehealth, network evolution, quality of service, billing support, operations, and more. These priorities follow a fast-track development lifecycle — from design and innovation through solutions that include standards, specifications, requirements, business use cases, software toolkits, and interoperability testing.

ATIS is accredited by the American National Standards Institute (ANSI). ATIS is the North American Organizational Partner for the 3rd Generation Partnership Project (3GPP), a founding Partner of oneM2M, a member and major U.S. contributor to the International Telecommunication Union (ITU) Radio and Telecommunications sectors, and a member of the Inter-American Telecommunication Commission (CITEL). For more information, visit < www.atis.org >.

The *Operational Opportunities and Challenges of SDN/NFV Programmable Infrastructure* was developed for the **Technical and Operations (TOPS) Council**.

Published by
Alliance for Telecommunications Industry Solutions
1200 G Street, NW, Suite 500
Washington, DC 20005

Table of Contents

Table of Figures

Table of Tables

Operational Opportunities and Challenges of SDN/NFV Programmable Infrastructure

1 Introduction

The development of technologies, including network function virtualization and Software-Defined Networking (SDN), builds on previous concepts such as active networks and programmable networks to increase the level of programmability within the infrastructure (communication, computation, storage, etc.).

This document will identify operational issues and opportunities associated with increasing programmability of the infrastructure. For example, this includes OSS/BSS impacts, reliability/fault detection, and administration, as well as maintenance issues over the network element, and service life cycles of IP-infrastructure-based network elements. The FG will also identify likely changes in operational procedures and staffing skill sets required to support increasing programmability.

The need for greater programmability is a common thread inherent in recent technology concepts such as SDN; Network Function Virtualization (NFV); evolution in device capabilities (e.g., due to Moore's Law); evolution in service concepts [e.g., Service Oriented Architectures (SOA) and cloud service delivery models (e.g., IaaS, PaaS)]; and evolving market expectations. This leads to the top down consideration of the range of impacts as the industry pivots towards supporting various forms of programmability. This pivot is an industry transition that is potentially more significant than previous technology transitions (such as that from circuit to packet technologies) because of the impacts on operational and service aspects.

In considering this transition between the traditional device centric, service-siloed infrastructure (A in Figure 1) and the emerging programmable infrastructures (B in Figure 1), there may exist more than one path from A to B based on the situation of that particular infrastructure operator. The goal is to identify use cases that articulate the vision of programmable infrastructure (with a particular emphasis on customer visible capabilities), motivate further work in the problem space, and help the industry understand the range of challenges and opportunities from the programmability pivot.

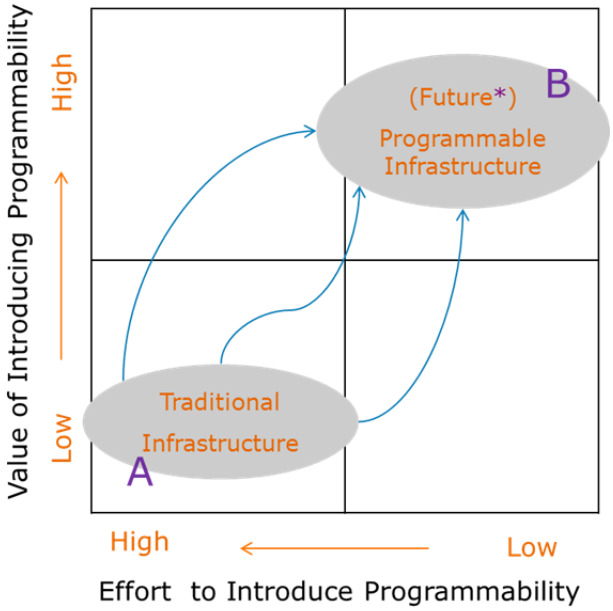

Figure 1: Value vs Effort of Introducing Programmability

2 Scope

The approach has been to conduct a top-down analysis of various programmability use cases leading to developing requirements for the following:

- Business Process Changes
- Information exchange between components
- Performance needs of components
- Personnel Staffing

Use cases and areas of standards gaps were identified as time permitted. There was *no plan for the development of technical solutions, protocols, and data models*. This document was completed in 6 months, ending in October 2013.

3 Programmability

Programs are generally considered as being comprised of algorithms operating on data structures[1]. There are various data structures associated with network infrastructures. For example, a given network element may implement data structures associated with:

- The protocols that the NE supports on its interfaces.
- The management information bases (MIBs) used to administer the NE (e.g., CLEI codes).
- Internal data structures used by software within the NE to perform the functions required of the NE.

An Operations Support System (OSS) may implement data structures associated with:

- The management information bases (MIBs) used to administer the NEs.
- The connectivity arrangements.
- Location of devices and facilities (e.g., CLLI codes).

A Business Support System (BSS) may implement data structures associated with:

- the subscribers/consumers of the network services (e.g., accounting records)
- business processes (e.g., work flow sequences).

Computer programming is the process of designing, writing, testing, debugging, and maintaining the source code of computer programs. Whatever the approach to development of software, the final program may be evaluated for various properties, including reliability, robustness, usability (ergonomics), portability, maintainability, and efficiency/performance. Programming languages can be used to create programs to execute algorithms or control the state of a machine. In the case of programmability of network infrastructure, the program may control the state of the network infrastructure, or be used to transform the data being transported through the network.

Programming paradigms are fundamental styles of computer programming, including imperative declarative, functional, and object-oriented paradigms. *Imperative programming* is a paradigm that

[1] See, Wirth, Niklaus (1976). *Algorithms + Data Structures = Programs (in English).* Prentice-Hall. ISBN 978-0-13-022418-7

presents a sequence of statements or instructions that change the state of a program. Low level programming languages present instructions using the instruction set of the processor (e.g., x86 instruction set). *Declarative programming* expresses what the program should accomplish without providing a sequence of actions to be taken. Functional (e.g., Lisp) or logical programming languages implement a declarative programming paradigm. *Functional programming* is a programming paradigm that treats computation as the evaluation of mathematical functions and avoids state and mutable data. *Logical programming* languages (e.g., Prolog) are based on first order logic (e.g., horn clauses). Programmability of network infrastructure may use different paradigms, languages, and tools for different purposes. The process of compilation, for example, transforms source code from high level languages into lower level languages such as executable instruction sets and can enforce certain rules -- e.g., enforcing data types. The research community has recently developed some more networking-centric programming paradigms, such as the data modeling language YANG[2], and the notion of a network operating system[3]. Network programming languages such as *frenetic*[4] are also evolving that can leverage SDN abstractions. Programmable approaches to design, configuration, and deployment of networks represent a considerable departure from current network element-centric operational practices.

There are a variety of approaches to the development of software, and assumptions about the software lifecycle. ISO/IEC 12207 can be used to describe software life-cycle processes[5]. Software development models include waterfall, spiral, iterative, agile, etc. DevOps is a software development method that is a response to interdependence between software developers and information technology operations. End User Development (EUD) describes activities and technologies that permit end users to create or modify software artifacts. Examples of EUD include scripting languages (e.g., Python), Programming by Example (PbE), configuration files that blur the lines between programs and data, workflow process models, and visual programming. Different software development approaches may be used when software is developed by vendors of software artifacts or network elements, infrastructure operators, and end users (whether consumer or enterprise). These agile, software-based approaches to design configuration and deployment of network services enable a significant change by the operator in degree and responsiveness of customer engagement (e.g., APIs and self-service portals in place of paper processes).

3.1 Operator Infrastructure Programmability in the Present Mode of Operation

The current operator infrastructure deployments are operator-specific in the architecture and locations where infrastructure is deployed and in the range of services offered. Historically, operator infrastructure has been comprised of proprietary hardware and software components that typically did not offer programmability by anyone other than the vendor of that equipment. Programmability of the operator infrastructure in the present mode of operation is only provided to the operator through the use of Operations Support Systems (OSSs) and Business Support Systems (BSSs). These OSS and BSS were developed to support Operations, Administration, Maintenance, and Provisioning (OAM&P) in various areas including Fault, Capacity, Accounting, Performance, and Security (FCAPS)[6]. FCAPS appears to have been replaced by the Fulfillment, Assurance, Billing (FAB) model popularized in eTOM[7]. The FCAPS model can be seen as bottom-up or network-centric in contrast with the FAB model, which looks at the processes in a top-down, customer/business-centric fashion.

---

[2] See <http://yang-central.org/twiki/bin/view/Main/WebHome>.

[3] See N. Gude et al, "NOX: Towards an Operating System for Networks," *Computer Communications Review* (2008) Vol. 38 pp. 105-110

[4] See < http://www.frenetic-lang.org/>.

[5] ISO/IEC 12207:2008 *Systems and software engineering — Software life cycle processes* < http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43447>.

[6] FCAPS is described in ITU-T, 1997, *M.3400 TMN management functions.*

[7] ITU-T, M.3050 *Enhanced Telecom Operations Map (eTOM) – The business process framework.*

Each operator maintains a unique set of OSS and BSS resources customized to support its business needs and adapted to its deployments of proprietary hardware and software components. A large operator may support thousands of instances of such OSS or BSS. While these OSS and BSS systems provide some degree of flexibility and programmability, in practice this is a very limited capability due to business constraints – e.g., maintaining support for legacy services.

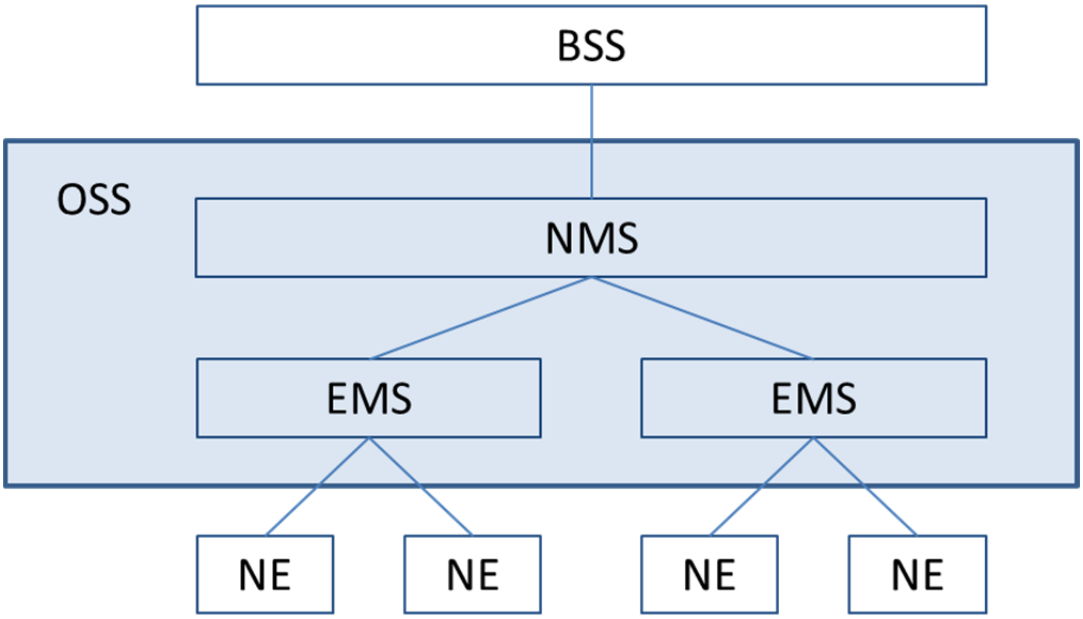

Figure 2: PMO OSS/BSS

Rather than focusing on an individual OSS or BSS system (which may or may not be of interest depending on the operator, service, infrastructure deployments, etc.), a framework providing a structured overview of OSS and BSS can provide a common vocabulary and concepts that are broadly applicable to the many different OSS and BSS. For the purposes of this document[8], the TM Forum Frameworx is adopted in this document a as a means to facilitate the discussion of the issues associated with these OSS and BSS.

There are four core frameworks provided in this suite of documentation: a Business Process Framework (eTOM), an Information Framework (SID), The Application Framework (TAM), and the Integration Framework as well as Business Metrics.

3.1.1 eTOM

eTOM provides a framework for business processes aligned with the Information Technology Infrastructure Library[9] (ITIL) that has been customized for typical operator business concern including marketing and offer management, service development and management, resource development and management, as well as supply chain development and management.

[8] The TM Forum claims that this suite of standards based tools and best practices has been adopted by 90 percent of the world's largest service providers. See TM Forum, *tmforum Case Study Handbook 2013*, page 6.

[9] ITIL 2011 has five volumes – Service Strategy, Service Design, Service Transition, Service Operation and Continuous Service Improvement.

3.1.2 SID

Consistency of data across the operators IT systems is facilitated by the use of Standardized Information Definitions (SID). These permit information to more easily flow across interfaces within the operator, as well as between the operator and its business partners and customers.

3.1.3 TAM

The Application framework provides a model for grouping processes and their related information into recognizable applications that span the multiple services that operators may offer from their infrastructure.

3.1.4 Integration Framework

The Integration Framework provides direction on how the operational processes can be automated using standardized information definitions (SID) and to define standardized Service Oriented Architecture (SOA) based management systems.

3.1.5 Business Metrics

A balanced scorecard of Business Metrics has been defined in areas such as:

- Revenue and Margin;
- Customer Experience; and
- Operational Efficiency.

Customer experience is not restricted to individual transactions on a single service, but may also include innovation in services as well. Open innovation may thus be seen as part of the enhancement of customer experience.

3.2 Operator Infrastructure Programmability in the Future Mode of Operation

The expectation of programmability is a common thread inherent in recent technology concepts such as: SDN; NFV; evolution in device capabilities (e.g., due to Moore's Law); evolution in service concepts (e.g., Service Oriented Architectures (SOA) and cloud service delivery models (e.g., IaaS, PaaS); and evolving market expectations. The expectation is that operators will continue deploying increasing qualities of generic computing nodes as part of their infrastructure. These computing nodes may be deployed in data centers, in central offices, in computing pods located elsewhere, as well as embedded in outside plant-deployed network elements and in mobile devices. As shown in Figure 3, SDN and NFV are expected[10] to enable open Innovation in operator infrastructures through the programmability created through the deployment of industry standard computing architectures.

The open innovation concept is a paradigm that assumes external ideas can and should be used by firms as well as internal ideas[11]. This is sometimes viewed as innovating with partners by sharing risks and rewards. However, it can also be interpreted as going beyond just using external sources of innovation, such as customer, rival companies, and academic institutions, to be a change in the development, use, management, and employment of intellectual property. In general, open innovation implies the systematic encouragement and exploration of a wide range of internal and external innovation which is then

[10] See *Network Function Virtualisation- Introductory White paper*. October 22, 2012.

[11] See H. Chesborough, *Open Innovation: The new imperative for creating and profiting from technology*, (2003), Boston Harvard Business School Press, ISBN 978-1578518371.

integrated with the capabilities of the firm and its resources to exploit those opportunities[12]. Open Source and open innovation might differ with respect to Intellectual Property Rights (IPR) issues, but are not mutually exclusive[13]. The European Commission also has an open innovation platform[14] as part of its policy initiatives in information society and media. Modern software engineering approaches embrace agile methodologies with their notion of rapid customer feedback, which can also be complementary to open innovation[15]. Open Innovation is about avoiding the groupthink[16] that may occur when pursuing closed innovation strategies. Concretely, programmability provides the framework and tools that allows a service consumer to "mash-up" existing services to create innovative functions. Vendor specific innovation can also occur within the framework of open source and open innovation.

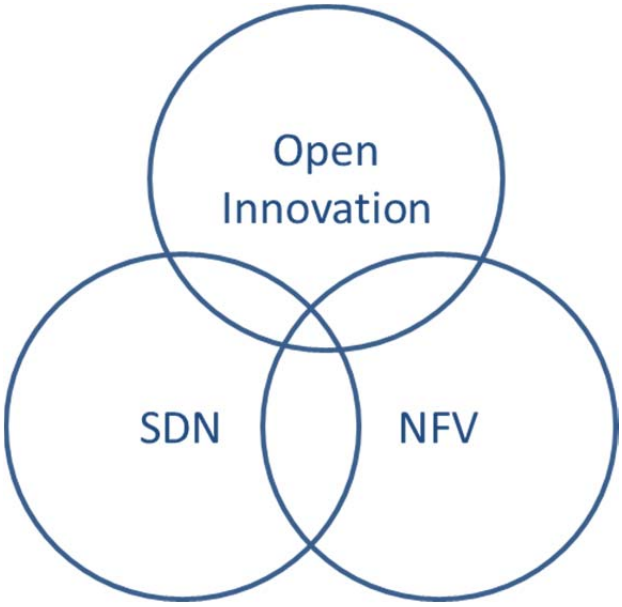

Figure 3: SDN and NFV as enablers for open innovation

The programmability of SDN/NFV technology will enable a dramatic reduction in the timeframe to introduce new network services. However, the expected OPEX saving may not be achievable, or the service may not even be deployable, if the necessary OSS/BSS functionality required to support the service cannot be realized in a similar timeframe, and manual procedures and process workarounds are required for an extended timeframe. Therefore, to realize those programmability benefits, modifications, or enhancements to existing OSS/BSS will be needed in parallel to the implementation of SDN, NFV, etc., in the network and control plane infrastructure. Alternatively, deployment of new OSS/BSS systems may be needed (or a hybrid combination of both approaches) to avoid long-term manual process workarounds.

---

[12] See J. West, S. Gallagher, *Challenges of open innovation: The paradox of firm investment in open-source software*, Rand D management 36 (3) 319 (2006).

[13] See IBM's eclipse platform – arguably an example of an open source project within an open innovation network. *Eclipse and Open innovation*,<Eclipse.org>, 12 Sept 2007.

[14] < http://files.openinnovation-platform.eu/polcydocs/open_innovation_2012.pdf>.

[15] See K. Conboy, L. Morgan, *Combining Open Innovation and Agile approaches: implications for IS project Managers* available online at :
<http://aran.library.nuigalway.ie/xmlui/bitstream/handle/10379/1424/Morgan_L_ECIS%20Paper.pdf?sequence=1 >.

[16] See J. Waclawsky, "Closed Architectures, Closed Systems, and Closed Minds", *Business Communications Review,* Oct. 2004.

In the PMO, Network Element based services are often managed in the interim via a vendor specific-EMS that accompanies the NE. A minimal EMS northbound (OSS) interface capability may be available at initial deployment, but manual procedures via CLI and GUI HMIs are often needed until full BSS/OSS interface capabilities are in place to fully automate the required business processes and operational workflows required to support the new service capabilities.

For the FMO, provider specific OSS/BSS interface requirements may still be needed. In addition, even in FMO, the software based network services may still need to interact with portions of the PMO BSS/OSS environment. Based on the Provider's BSS/OSS environment, an SDN/NFV service may need to include support for

- A "virtual EMS" capability that can provide local workcenter HMI access, and interfaces to a mix of current and next generation BSS/OSS systems; or
- A set of direct HMIs (CLI, GUI, etc.) and a set of direct provider specific BSS/OSS interfaces (with no EMS level functionality involved, virtual or otherwise); or
- A combination of the above.

Supporting the above range of possible BSS/OSS environments will require careful attention to software modularity (separation of concerns, layering, etc.).

Table 1 is a list of salient operational attributes and features of a potential future, programmable, -SDN-enabled infrastructure (FMO), in contrast to the current non-programmable infrastructure (PMO).

Table 1: Contrasting operational attributes and features of FMO and PMO

PMO	FMO
Manually ordered and provisioned services	Portal or cloud-triggered services, automatically instantiated
Static configuration, long lifetime services	Dynamic configuration/reconfiguration, long and short lifetime services
Network largely independent of applications, except specific Telco services	Network openly programmable by applications, including 3rd party applications
Low-moderate transaction volume	Very high transaction volume, driven by dynamic cloud apps and virtual network functions (VNFs)
Network abstraction by layer/domain, exposed for managing network	Integrated multi-layer(L3-L0)/multi-domain (access, metro, core) network abstraction, exposed via APIs to applications
Fragmented control and resource management, usually fragmented domain-by-domain and often even vendor-by-vendor	Centralized, multi-layer/multi-domain control and resource management
Policy-based access control and QoS	Policy-based end-to-end networking (connectivity, virtualization, multiple flow control points, etc.), in addition to access/QoS)
Distributed control plane	Mix of centralized and distributed control planes
Flow-level controls at selected policy enforcement points	Fine-grained flow-level controls at multiple points across the network
Hardware/firmware centric devices managed	Software centric network abstractions managed
Separate IT/Data Center and Network CO with hardware equipment from different vendors	Common technology and technical plant with predominantly virtualized software
Periodic (typically quarterly) software releases	Continuous software process –- "sandbox"; good integration with DevOps
Geographically fixed, single purpose equipment	Highly dynamic and configurable topology & roles
Tight coupled NE instance, generic EMS & NMS/OSS	Separation of physical and logical components

Separation of service elements and support systems	Integrated orchestration , automation, and virtualization
Faults as service failures; fault detection is pseudo real-time	Faults as capacity reduction events; real-time fault detection
Hardware based monitoring; need custom hardware	Software and probe based monitoring; no need for custom hardware
Service specific resource combinations	Profiles, templates, and reusable resource combinations
Special design and provisioning processes	Configurable catalog/ rule driven delivery frameworks
Optimized around provider networks and ops process	Optimized for customer experience

As Table 1 illustrates, there are a number of significant changes expected to affect the operational environment with the introduction of programmable technologies such as NFV and SDN. These technologies can also be deployed in ways that respect existing network architecture standards and interfaces. The FMO seems likely to emerge from component standards around APIs that the ecosystem finds useful rather than system level specifications[17].

3.2.1 Network Function Virtualization

NFV concepts are currently being developed through the ETSI NFV ISG, among others. The material in this section should be considered as preliminary and subject to change, as the ETSI NFV ISG reaches consensus and publishes its NFV documents.

NFV introduces the concept of transitioning operator infrastructure from the existing siloed, proprietary hardware components to a model of software components (Virtualized Network Functions –- VNFs) running on generic computing elements as the target infrastructure for supporting current and future operator services. The generic computing nodes within the NFV Infrastructure (NFVI) provide a platform for increased programmability of the operator infrastructure.

[17] See J. Waclawsky, “Where do system standards go from here?”, *Business Communications Review*, March 2005.

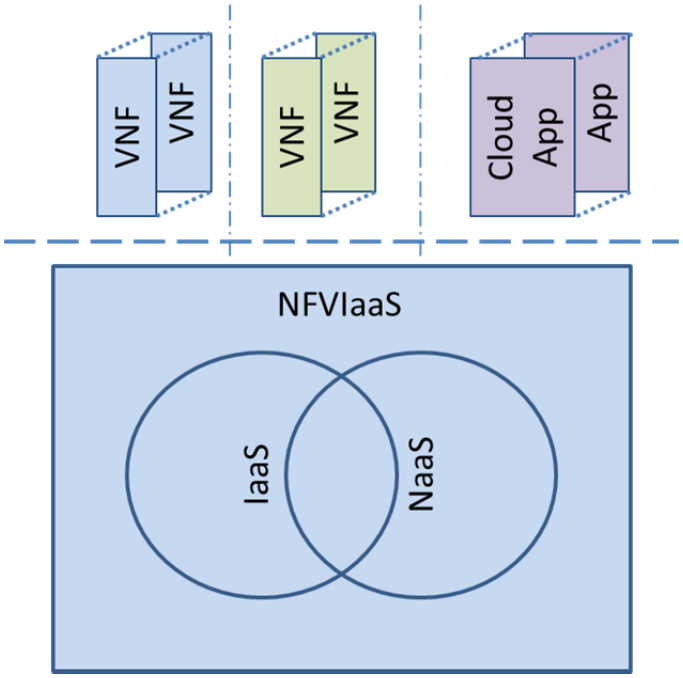

Figure 4: Network Function Virtualization

Figure 4 shows the NFVI providing both general purpose computing infrastructure services (IaaS) as well as dynamic connectivity services (NaaS). SDN is expected to provide capabilities required for NaaS. The applications that can be supported by this NFVI include both general cloud applications as well as VNFs. The administration of the NFVI and the applications that run on that infrastructure could be independent (e.g., different departments within the same operator). The applications running on the NFVI could be administered independently (indicated by the different colors –- blue, green, and purple in Figure 4).Each administrative region (indicated by dashed lines in Figure 4) could support multiple instances of the same or different applications (indicated by dotted lines in Figure 4).

3.2.2 SDN

SDN concepts have been developed initially at the Open Networking Foundation (ONF), with the OpenFlow Specifications. Depending on the definition of SDN that you adopt, various other organizations are developing or have developed SDN relevant protocols (e.g., IETF's i2rs effort).

SDN is a technology or architecture that is continuing to form and mature. As a result of our study, the following elements were identified as the most relevant:

- Centralization of control;
- Programmability of multiple network layers/network traffic via APIs;
- Control-forwarding/data plane decoupling;
- Control plane to a virtual or physical network;
- Support of multiple, isolated virtual networks; and
- Complementary relationship with network virtualization/abstraction.

The term SDN is commonly associated with the partitioning of device architectures into separate components for processing the data and control plane aspects, as illustrated in Figure 5. The control plane aspects can then be logically centralized in an SDN controller element (or elements) implemented on generic computing infrastructure. SDN controllers provide a mechanism to introduce more flexible programmability into the operator's infrastructure, compared to conventional monolithic data/control plane

elements and fully distributed control planes. This view of SDN focuses on northbound interfaces from the network elements.

For operator networks, arguably the greatest challenge to making networks programmable lies in the upper control and management layers of the network, where new and legacy systems must co-exist through evolving stages, and fragmented domain-by-domain control and management must be migrated into an integrated whole in order to achieve seamless automation. As a result, programmability will necessarily need to be phased into the network, and it is important to prioritize on the most beneficial aspects of network programmability first. Therefore, this ATIS whitepaper focuses on the top-down aspects –- both benefits and challenges –- of network programmability, through the analysis of top-down use cases.

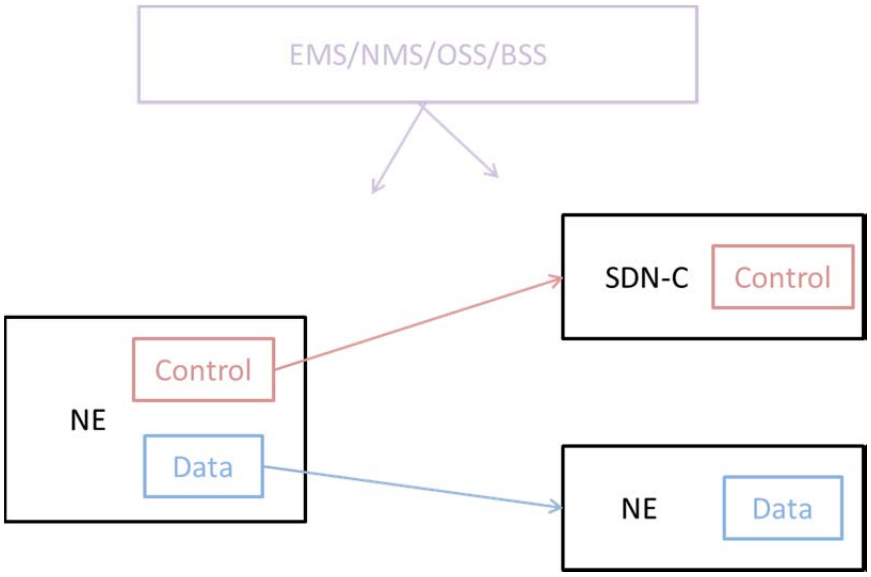

Figure 5: Separation of Control and Data in SDN

The existing NE is controlled by an EMS/NMS/OSS/BSS in the PMO environment. The introduction of SDN introduces a new component – the SDN Controller (SDN-C in Figure 5), and potentially reduces the functionality of the NE. Changes in the functionality of the NE require corresponding changes in the EMS/NMS/OSS/BSS. Introduction of a new type of NE for the SDN-C would require additional capabilities in the EMS to model that device, while some roles of the EMS/NMS/OSS/BSS would likely be subsumed by the SDN-C. On the right side of the above figure, the EMS/NMS/OSS/BSS controls both the SDN controller and the network element. Alternatively, some of the EMS/NMS/OSS/BSS and SDN-C functions could be merged.

The control plane can be distributed between the SDN Controller and network elements, enabling a hybrid SDN model. This is the model advocated by IETF I2RS where work is in progress.

For an alternate view, consider the conceptual model of SDN architecture in Figure 6 adapted from the ODCA[18]. In this architectural view the OSS/BSS functions and services (e.g., DDoS prevention) are merged in an application plane that communicates through a northbound API with the SDN controller.

[18] Open Data Center Alliance (ODCA) *Usage model: Software-Defined Networking* Rev1.0 (2013) , Fig ,1 pg 9.

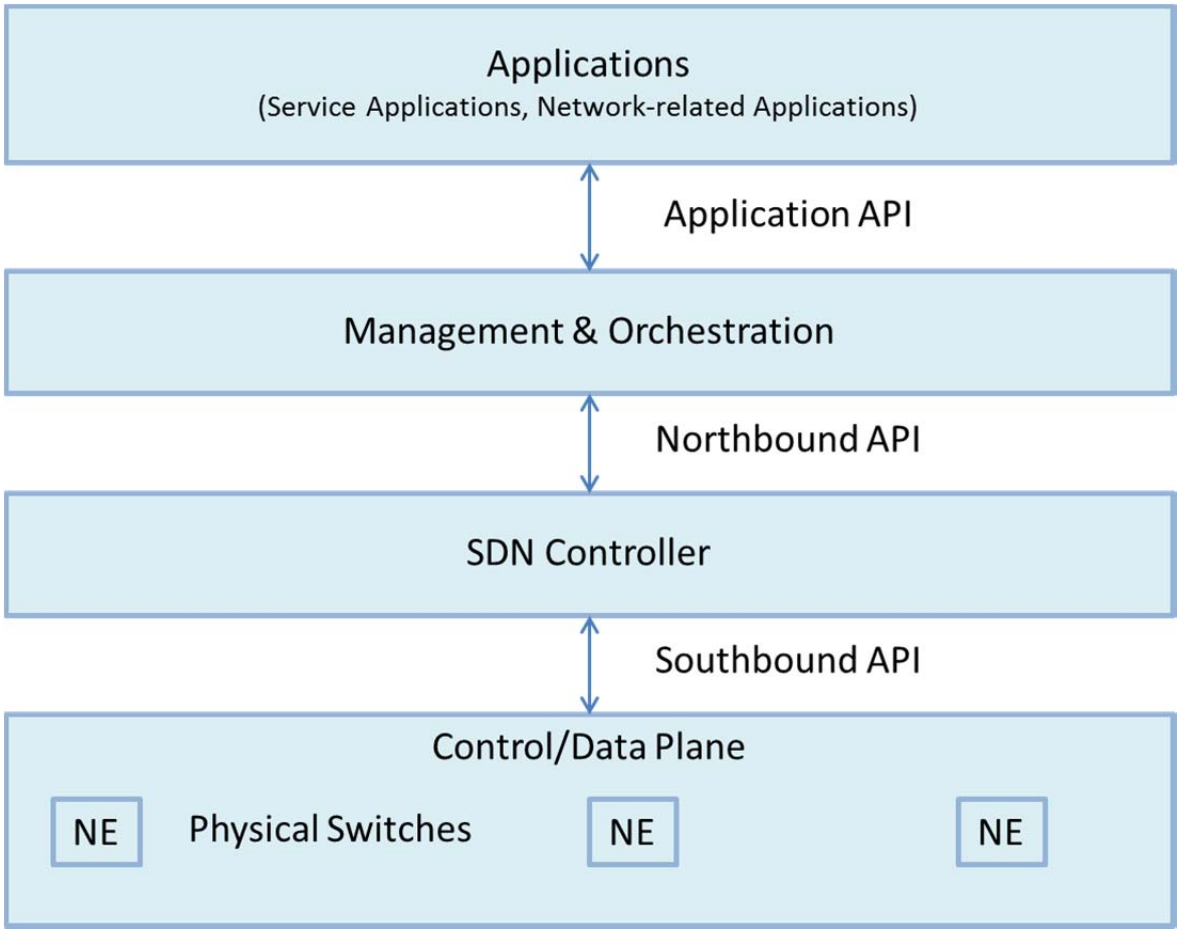

Figure 6: Conceptual Model of SDN Architecture (adapted from ODCA)

3.2.3 Cloud Services

Operators are increasingly providing cloud-based services to external parties, as well as using them for internal operations. Cloud services provide programmable resources in a variety of service models – e.g., IaaS, PaaS, SaaS.

3.2.4 Support for Big Data Analytics & Network Optimization

In addition to adding programmability via SDN or other techniques, consideration should be taken as to how to increase the interaction between the data embedded in a network and the systems stacks driving the network. Today, we have a fairly passive and limited interaction with the data contained in the network, it is polled typically using 'SNMP gets'/bespoke scripts and a limited set of information sent to the NMS, using 'SNMP traps' for example. In the operating environments of today these data are usually distributed across an organization, with little to no correlation. Access to the network for systems which poll/collect information is controlled by often complex security policies. When considering the addition of programmatic functions in an NE, and an OSS which is able to use these features, enabling 'feedback' becomes something to consider. A feedback loop can be best described as where the network is actively relaying data back to the OSS, which in turn digests this information as part of a workflow to actively manipulate either a single network element or multiple networks, for the purposes of network optimization, for example. While an endless number of events/actions can be conceived of, they can best be described as persistent, non-persistent, and ephemeral (momentary). Depending on the type of action an OSS determines, the OSS might take one or more of the three types of actions. One of the key enablers for this type of activity is programmatic capability. The current model of static configuration (persistent), both

at the device level and at the OSS level, does not easily allow for non-persistent and ephemeral manipulation of the network. As part of a feedback loop, external tools/plugins to an OSS can be used to enhance the decision making processes, with information received from the network and from external sources.

4 Use Cases

4.1 Overview

There is a potentially infinite set of use cases that might be considered, but for the purposes of the SDN-FG, the most value is achieved in the time available by selecting a set of use cases to explore and illustrate the programmability problem space. The use cases here provide a mechanism to explore and illustrate the problem space with technically feasible and reasonably realistic uses, rather than any judgment regarding market viability or particular service examples.

One approach to explore the problem space is to partition it in terms of ***who owns*** the programmable infrastructure and ***who programs*** it. Figure 7 provides a simple reference model to consider the impacts of programmability on the infrastructures on Operators as well as Enterprises and Consumers. Operator infrastructures are assumed capable of supporting multiple instances of both Consumer and Enterprise Infrastructures; Figure 7 shows them from separate operator infrastructures for diagrammatic convenience. The Operator Infrastructures are assumed to support multiple types of services, including: wireless, wireline, content, Public Cloud, and operator infrastructure based or decoupled over the top services. Two instances of Operator Infrastructure are shown in order to illustrate possible network interconnection use cases between Infrastructure Operators A and B. The Consumer and Enterprise Infrastructures are assumed to include capabilities for mobile access and multiple locations. An Enterprise Infrastructure may include one or more data centers as well as mobile devices. A Consumer Infrastructure may include mobile devices and home network elements in one or two locations.

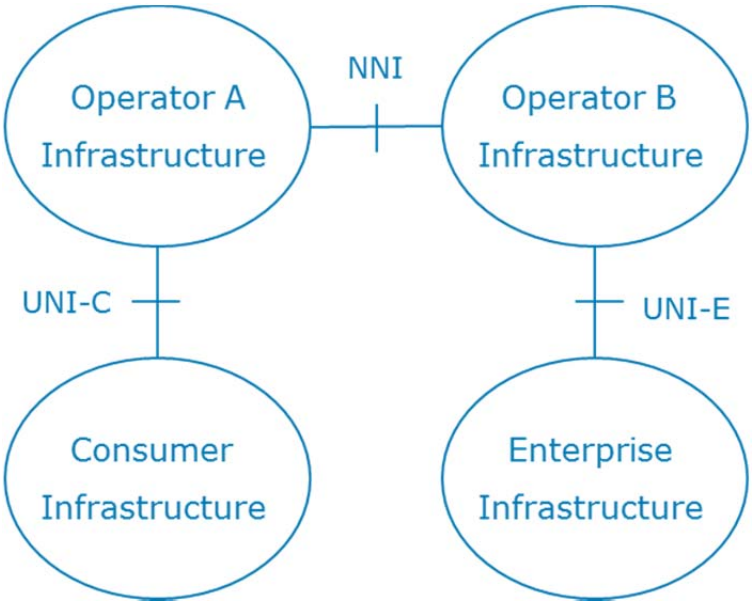

Figure 7: **Infrastructure Entities**

With three infrastructure entities identified (Operator, Enterprise, Consumer), there is a limited number of permutations to be considered. The below table illustrates these nine permutations. These nine permutations can then be used to provide a context to identify a few use cases to explore and illustrate various impacts from programmability within that context.

Table 2: Permutations of Infrastructure Entities

Owns \ Programs	Operator	Enterprise	Consumer
Operator	Internal (A-A) or NNI (A-B)	UNI-E	UNI-C
Enterprise	UNI-E	Internal	Operator Supported?
Consumer	UNI-C	Operator Supported?	Internal

4.2 Operator Programs Operator Infrastructure

4.2.1 Operator A Programs Operator A Infrastructure (Internal)

- *Fault detection / isolation / recovery & sparing Use Case*
  - Navigating the layers of Tunnels, Overlays, and Abstractions.
- *New Service Type Introduction / Open innovation Use Case*
  - Validation, regression, system integration, DevOps, OSMINE.
- *Maintenance on operational infrastructure NEs running 3rd party code Use Case*
  - Non data center application, multivendor interoperability/isolation.

4.2.1.1 Service Provider "DevOps"

4.2.1.1.1 Service/Application Development & Test Use Case

Title	**DevOps: Service Development and Test**
Actors	**SP DevOps team.** **Service execution test resources.** **Support test network resources.**
What is Programmable?	**A service development and test environment including the supporting network. This use case covers only testing purposes.**
Description	**A brand new service (or application), or service feature is installed in the service provider's facilities but in a protected "sandbox" environment that is suitable for realistic testing. The sandbox consists of two major areas: the application run time environment made up of compute and storage resources, and a network that connects various run time resources together.** **This environment is isolated in a way that the production applications and network are not impacted regardless of the new service's behavior.** **The setup is automated such that from configuration information, the sandbox environment can be instantiated on demand. Further, the execution of test cases within the environment is also automated via scripts or configuration.** **This use case focuses on the network portion of the sandbox environment and interface to the service execution resources.**

	In addition to testing new services or service features, an existing deployed service may also be tested with different configurations. In other words, testing scenarios may not always require software changes on the service side.
Traditional Approach	**Although software development (write, compile, build) may be already automated, installing each code modification requires a number of configuration changes to the supporting network (sometimes across many config files, sometimes manually using CLIs on various pieces of equipment). Further, test case execution also requires some manual intervention and supervision.**
Programmability Approach	**"Developing a new service" and "testing it" become unified in terms of hardware, software, test cases, configuration, and provisioning. Software updates may be programmatically tested in the network in an automated manner.**
Network Programmability	**A programmatic interface exists between the service execution and the supporting network. The supporting network is instantiated with the following:** • **Static network configuration in terms of topology, addressing, basic capabilities (L2 or L3 operation), supporting network services (naming, discovery, security), etc.** • **Current capacity requirements on each network link (if required by service) in terms of QoS parameters. Reservation of these capacities.** • **On demand changes to any of the items set during the execution of the service. This includes topology and capacity changes (perhaps within some pre-arranged limits). These changes could be installed via SDN.** • **Optionally, there is some response from the network to the service and potentially some negotiation if sufficient resources are not available.** • **Some feedback from the network to the service during service execution for any errors or unforeseen events.**
Transition Challenges	**The network must be able to engage the necessary physical resources needed to support the service on demand and in an automated fashion.** **The network must be able to setup the test network in an isolated way so as not to impact the production network.** **Service Providers may now get involved with software development to some degree in order to modify or customize solutions delivered by vendors.**
Advantages	**Experiment with new untried services and measure their performance and impact to providers' network.** **Make rapid incremental changes to test improvements in services without impacting production network.** **Gives the provider the ability to customize services.**

4.2.1.1.2 Service/Application Development & Test Northbound Requirements

The following are the extracted requirements and descriptive behaviors of the programmable interface between the Service Provider service development and test environment and the application run time environment based on the use case. Here the focus is on the network portion of the run time environment

- A protected "sandbox" run time environment shall be instantiated on demand in an automated fashion by the service development and test environment.

- All parameters of the service shall be exercised through this interface by the service development and test environment. This enables a full suite of test cases to be automated.
- Service logic (software) may be changed and executed in the run time environment, configuration information may be changed and executed in the run time environment, or a combination of both. This supports all testing scenarios.
- The interface should allow the run time environment to be instantiated with static network configuration in terms of topology, addressing, basic capabilities (L2 or L3 operation), supporting network services (naming, discovery, security), etc.
- The interface shall allow the run time environment to be instantiated with current capacity requirements on each network link (if required by service) in terms of QoS parameters. At the same time the reservation of these capacities in the network is also necessary.
- The interface allows the run time environment to be modified with on demand changes to any of the items set during the execution of the service. This includes topology and capacity changes (perhaps within some pre-arranged limits).
- The interface shall allow some response from the network to the service and potentially some negotiation if sufficient resources are not available.
- The interface should allow some feedback from the network to the service during service execution for any errors or unforeseen events.

4.2.1.1.3 Service/Application Release & Deploy Use Case

Title	DevOps: Service/Application Release and Deploy
Actors	SP DevOps team. Service execution resources. Support test and production network resources.
What is Programmable ?	A service execution run time environment including the supporting network. This use cases covers only release and deployment.
Description	A companion use case to “Service/Application Development and Test”, this looks at the next steps of moving changes into a production environment and releasing it for commercial use. The production environment consists of two major areas: the application run time environment made up of compute and storage resources and a network that connects various run time resources together. This use case focuses on the network portion of the run time environment and interface to the service execution resources. After a new service (or application), service feature, or just a set of new configuration changes are properly vetted, the changes must be moved into the production environment. This use case covers the installation and activation of new software as well as modifications of existing software (updates, patches, etc.). This use case is part of a greater process the SP follows to track and manage releases on a continuous basis. Note that configuration changes are treated the same as software changes. That is, a new release might consist of only configuration changes, only software changes, or both. Specifically, software and configuration changes have first been tested in a safe “sandbox” run time environment. These changes must then be moved into a production environment in an automated manner as much as possible to avoid introducing any new errors. Further, the process needs to support release planning and tracking so that future releases can be deployed and adjusted based on business needs, performance, and customer feedback.

Traditional Approach	**Although software deployment and release tracking is already widely practiced, installing each code or configuration modification currently requires a number of manual steps (sometimes across many configuration files, sometimes using CLIs and scripts on various pieces of equipment). So while software development might be automated and employ modern tools, deployment in the run time networking environment is far from automated and requires a lot of manual intervention, planning, and supervision, thus incurring time delays and costs.**
Programmability Approach	**With the right programmable interfaces, releases and deployment can be a process integrated with develop and test. Software updates may be programmatically installed in the network in an automated manner.**
Network Programmability	**A programmatic interface exists between the Service Provider service development and test environment and the network portion of the run time environment. This interface supports:** • **Carrying over or duplicating the static network configurations as well as the network requirements of each link (if required by service) in terms of QoS parameters and network resource reservations from the test sandbox environment into the production network environment instance.** • **For new services, installing new network instances and making them available for commercial use.** • **For existing services, migrating commercial service from the previous release instance to the new release instance. This deployment of a new release must minimize any service disruptions as required by the specific service(s).** • **Ability to track each release and manage multiple simultaneous releases which might be in commercial service.** • **On demand changes to any of the items set during the execution of the service. This includes topology and capacity changes (perhaps within some pre-arranged limits). These changes could be installed via SDN.** • **Optionally, there is some response from the network to the service and potentially some negotiation if sufficient resources are not available.** • **Some feedback from the network to the service during service execution for any errors or unforeseen events.**
Transition Challenges	**The network must be able to engage the necessary physical resources needed to support the old service release and transition to the new release in an automated fashion.** **Service Providers now need to spend efforts on automating and maintaining this release and deploy process instead of the manual methods used previously.** **This use case increases the number and frequency of releases to be handled by service providers.** **Get involved with software development to some degree in order to modify or customize solutions delivered by vendors.**

Advantages	**Have higher confidence that production deployments will work correctly.** **Make rapid incremental changes to production services at much lower costs.** **Do release planning more effectively.** **Gives the provider the ability to customize services.** **Automated orchestration deploying the new services avoids human error during deployment.**

4.2.1.1.4 Service/Application Release &Deploy Northbound Requirements

The following are the extracted requirements on the programmable interface between the Service Provider service release and deploy environment and the application run time environment. Here the focus is on the network portion of the run time environment:

- Support for the installation and activation of new software as well as modifications of existing software (updates, patches, etc.).
- Service logic (software) may be changed, configuration information may be changed and executed in the run time environment, or a combination of both. This supports all forms of releases.
- The interface supports carrying over or duplicating the static network configurations as well as the network requirements of each link (if required by service) in terms of QoS parameters and network resource reservations from the test sandbox environment into the production network environment instance.
- For new services, the interface allows installing new network instances and making them available for commercial use.
- For existing services, the interface allows migrating commercial service from the previous release instance to the new release instance. This deployment of a new release must minimize any service disruptions as required by the specific service(s).
- The interface allows the ability to track each release and manage multiple simultaneous releases which might be in commercial service.
- The interface allows the run time environment to be modified with on demand changes to any of the items set during the execution of the service. This includes topology and capacity changes (perhaps within some pre-arranged limits).
- The interface allows some response from the network to the service and potentially some negotiation if sufficient resources are not available.
- The interface allows some feedback from the network to the service during service execution for any errors or unforeseen events.

4.2.1.2 Flow Aware Networking: Large Flow LAG Load Balancing

4.2.1.2.1 Flow Aware Networking: Large Flow LAG Load Balancing Use Case

Title	Flow Aware Networking -- Large Flow Link Aggregation Group (LAG) Load Balancing
Actors	Network Operator.
What is Programmable?	Switches and Routers in the Operator Network.
Description	This use case addresses the current issues in hash based LAG load balancing by dynamically programming the network to steer large flows and achieving optimal egress link utilization.
Traditional Approach	Uses static hash based load balancing.
Programmability Approach	Large flow detection/load-balancing SDN application in open source (e.g., OpenDaylight) framework which does the following • The Large flow detection/load-balancing SDN application identifies congested egress links for LAG in a switch/router and performs the following: o Alternative placement of large flows – ingress PBR o Redistributing other flows -- adjust LAG hash table • Optimal egress link utilization for LAG is thus achieved. More details are in section 4 of the IETF draft below: <http://datatracker.ietf.org/doc/draft-ietf-opsawg-large-flow-load-balancing/>
Transition Challenges	Some parts of the network, specifically LAG ports in switches/routers, are being controlled using a combination of centralized and traditional approaches (Hybrid SDN model) while some other parts of the network are being controlled using traditional switching/routing. This would introduce challenges especially in the area of troubleshooting. Some of the southbound requirements (LAG table programmability) is yet to be standardized in Openflow; the mitigation plan would be to use vendor specific southbound plugins in open source (e.g., OpenDaylight) framework.
Advantages	*Capex reduction*: WAN/DC core bandwidth savings with LAG. *Facilitate 10G to100G WAN migration strategy*: Bundling links of different speeds. [19]

4.2.1.2.2 Flow Aware Networking: Large Flow LAG Load Balancing Southbound Requirements

Information Model requirements for Southbound API in OpenDaylight framework (combination of new and existing requirements):

- Need standard LAG table interface -- Openflow (new).

[19] IETF composite link reference: <http://datatracker.ietf.org/doc/draft-ietf-rtgwg-cl-use-cases/ >

- Link Utilization – IETF (existing):
  - For normal speed links, use Interface table (iftable) MIB [RFC 1213]; For high speed links, use etherStatsHighCapacityTable MIB [RFC 3273].
  - For further scalability, it is recommended to use the counter push mechanism in [sflow-v5] for the interface counters; this would help avoid counter polling through the MIB interface.
- Large flows which are recognized in the SDN application.
  - All fields in the packet which can be used for making forwarding decisions – Openflow (existing).
    - LAG member port in flow table response for large flow steering – Openflow (new).
- More Details: IETF Working group draft – section 5, Information Model <http://datatracker.ietf.org/doc/draft-ietf-opsawg-large-flow-load-balancing/>.

4.2.1.2.3 Flow Aware Networking: Large Flow LAG Load Balancing Northbound Requirements

Information Model requirements for Northbound API in OpenDaylight framework (only new requirements):

- Alternative Placement of large flows.
  - All fields in the packet which can be used for making forwarding decisions.
  - LAG member port for large flow steering of L2/L3 traffic.
  - Nexthop IP/MAC address for large flow steering of L3 traffic.
- Redistribution of small flows.
  - Weight of each member in LAG table.
- Monitoring (other applications like EMS can take advantage of this).
  - Number of times rebalancing was done for the router/switch.
  - Time since the last rebalancing event for the router/switch.

4.2.2 Operator A Programs Operator B Infrastructure (NNI)

4.2.2.1 Network Function Virtualization Infrastructure as a Service (NFVIaaS)[20]

Title	Network Function Virtualization Infrastructure as a Service (NFVIaaS)
Actors	**Service Provider #1.** **Service provider #2.**
What is Programmable?	**Both Service Provider #1 and Service Provider #2 have deployed a Network Function Virtualization Infrastructure (NFVI). The NFVI provides capabilities comparable to an IaaS cloud computing environment coupled with a NaaS that can provide on demand connectivity within some defined set of service termination points. The NFVI is a programmable infrastructure for authorized users.**
Description	**By agreement between the two service providers, Service Provider #1 is able to instantiate its own VNFs on the NFVI operated by Service Provider #2. Using the VNFs running on both NFVIs, Service Provider #1 can deliver end-end services across both NFVIs.** **Since VNFs are not all location independent, this capability is expected to improve latency and reliability. It also enables compliance with some regulatory constraints on the location of processing and storage.**

[20] The NFVI concept originated with the ETSI NFV ISG. The material in this section provides a preliminary view and may change as the ETSI NFV ISG reaches consensus and publishes its NFV documentation.

Traditional Approach	**The traditional approach is a composite of manually, specially engineered wholesale services.** **Service Provider #1 would need to negotiate individual leased lines, cloud computing resources, etc. with Service provider #2. Assembling, and then operating the end-end service would be manually intensive and require significant coordination between the service providers – e.g., regarding trouble shooting and maintenance.**
Programmability Approach	**By agreement, Service Provider #2 authorizes Service Provider #1 to operate on NFVI that is operated by Service Provider #1. The NFVI provides the NaaS capabilities to dynamically connect to the authorized termination points within the NFVI. The NFVI provides the compute nodes with IaaS capabilities to provide the run time environment for the VNFs to be installed by Service Provider #1. Service Provider #1 gets real-time updates on the status of the NFVI and is able to remedy malfunctions without the involvement of Service Provider #2.**
Network Programmability	**This IaaS environment is a generic computing platform programmable by authorized users. In this case, Service Provider #1 is an authorized user. The NaaS provides on demand connectivity with authorized termination points within the NFVI. SDN based separation of data and control plane functions are expected to be an important enabler for scaling the NaaS capabilities.**
Transition Challenges	**Not all network functions can be economically virtualized and so the NFVI must interface to existing physical network elements.** **Deploying the NFVI impacts the existing OSS and BSSs which must be extended to support the VNF equivalents of existing Network Elements, as well as the orchestration of all these VNFs and the NFVI into an end to end service.** **Network operations based around physical connections become refocused on APIs and flows. Purpose built hardware becomes replaced by software defined VNFs. Manual deployments become replaced by automated orchestration of workloads. Sites with specialized NEs become replaced by cloud technology pods with generic computing infrastructure as the computer nodes of the NFVI.** **Operator staff skill sets have to change to support a much more software intensive operations paradigm.** **Operator business practices have to support commercial offers of NFVIaaS as wholesale offers between operators with all the required details of service level agreements and other terms and conditions.**
Advantages	**The NFVI provides a shared infrastructure with pooled resources that can by dynamically allocated between different VNFS and other cloud computing workloads providing cost advantages over infrastructures dedicated to individual (siloed) services.** **NFVIaaS provides full control of the computing resource through administrative access to the VMs while trading the capital and operational cost of that computing infrastructure for a flexible, efficient rental of the capacity required. The NFVIaaS control of the infrastructure enables easier portability and interoperability of services developed by Service Provider #1 in their own NFVI.** **The ability of Service Providers to rapidly deploy services across each other's infrastructure should enable more rapid response to enterprise customer requests for service reconfigurations, etc.**

4.3 Enterprise Programs Operator Infrastructure

4.3.1 Operator exposes Virtual Network Platform as a Service (VNPaaS[21]) to Enterprise

Title	Operator exposes Virtual Network Platform as a Service (VNPaaS) to Enterprise
Actors	Service Provider. Enterprise.
What is Programmable ?	The Service Provider has deployed a Network Function Virtualization Infrastructure (NFVI) in order to provide a PaaS offer. The NFVI provides capabilities comparable to an IaaS cloud computing environment coupled with an NaaS that can provide on demand connectivity within some defined set of service termination points. The NFVI is a programmable infrastructure for authorized users. The Service Provider bundles a number of NFVI capabilities together to offer a platform for the Enterprise to create its own virtual networks within the Service Provider's NFVI
Description	By agreement between the Enterprise and the Service Provider, the Enterprise is able to instantiate its own VNFs on the NFVI operated by the Service Provider. Since VNFs are not all location independent, this capability is expected to improve latency and reliability. It also enables compliance with some regulatory constraints on the location of processing and storage. The combination of IaaS and NaaS capabilities in the Virtual Network Platform offered by the Service Provider permits the Enterprise to create, for example, a virtual data center.
Traditional Approach	The traditional approach is a composite of manually, specially engineered wholesale services. Enterprise would need to negotiate individual leased lines, cloud computing resources, etc., with multiple Service Providers. Assembling, and then operating the end-end service would be manually intensive and require significant coordination between the service providers – e.g., regarding trouble shooting and maintenance.
Programmability Approach	By agreement, Service Provider authorizes Enterprise to operate a PaaS offer on NFVI that is operated by Service Provider. The VNPaaS includes higher-level abstractions of networking and computing elements. The NFVI provides the NaaS capabilities to dynamically connect to the authorized termination points within the NFVI. The NFVI provides the compute nodes with IaaS capabilities to provide the run time environment for the VNFs. Service Provider gets real-time updates on the status of the NFVI and is able to remedy malfunctions without the involvement of Enterprise.
Network Programmability	This PaaS environment is a computing platform programmable by authorized users with higher level abstractions than an IaaS. In this case Service Provider is an authorized user. The NaaS provides on demand the connectivity with authorized termination points within the NFVI. SDN based separation of data and control plane functions are expected to be an important enabler for scaling the NaaS capabilities.

[21] VNPaaS concept originated with the ETSI NFV ISG. The material in this section provides a preliminary view and may change as the ETSI NFV ISG reaches consensus and publishes its NFV documentation. See GSNFV 009 (2013-10), *Network Function Virtualisation: Use cases*.

Transition Challenges	**Not all network functions can be economically virtualized and so the NFVI must interface to existing physical network elements.** **Deploying the NFVI impacts the existing OSS and BSSs, which must be extended to support the VNF equivalents of existing Network Elements, as well as the orchestration of all these VNFs and the NFVI into an end to end service.** **Network operations based around physical connections become refocused on APIs and flows. Purpose built hardware becomes replaced by software defined VNFs. Manual deployments become replaced by automated orchestration of workloads. Sites with specialized NEs become replaced by cloud technology pods with generic computing infrastructure as the computer nodes of the NFVI.** **Operator staff skill sets have to change to support a much more software intensive operations paradigm.** **Operator business practices have to support commercial offers of VNPaaS with all the required details of service level agreements and other terms and conditions.**
Advantages	**The NFVI provides a shared infrastructure with pooled resources that can be dynamically allocated between different VNFs and other cloud computing workloads providing cost advantages over infrastructures dedicated to individual (siloed) services.** **VNPaaS provides limited control of the abstractions provided by the virtual network platform while trading the capital and operational cost of that computing infrastructure for a flexible, efficient rental of the capacity required. The VNPaaS control of the infrastructure enables easier portability and interoperability of services developed by Service Provider.** **The ability of the Service Provider to rapidly deploy services should enable more rapid response to enterprise customer requests for service reconfigurations, etc.**

4.4 Consumer Programs Operator Infrastructure

Because of the wide variation of in the programming skill levels of consumers, “SaaS-like” offers where the degree of programmability is limited to some configuration options rather than general purpose programming languages may enable a wider audience to utilize the service.

4.4.1 Consumer Uses Service Provider’s “SaaS-Like” Offer of a Firewall Service

Title	**Consumer uses Service Providers “SaaS-like” offer of a firewall service**
Actors	**Consumer.** **Service Provider.**
What is Programmable?	**The Service Provider offers a firewall function on a “SaaS-like” basis with various options configurable on the firewall.**

Description	**Consumer subscribes to an offer by a Service Provider for a firewall service.** **The Service Provider deploys the firewall capability within its infrastructure.** **Network traffic to/from the Consumer's devices are routed through the firewall.** **The service offer provides a facility for the Consumer to configure various firewall options (e.g., opening and closing pinholes).**
Traditional Approach	**Traditional firewalls are deployed as standalone elements, or integrated into CPE such as residential gateways. Consumer configuration of traditional firewalls is typically difficult and requires specialized knowledge. Network-based firewall configuration typically does not provide the consumer with programmatic control.**
Programmability Approach	**The programmability offered to the consumer is likely limited to some set of configurable features in order to better match market expectations. The programmability may be achieved here by selecting a few buttons on a web page offered by the Service Provider (after suitable authentication).**
Network Programmability	**The firewall could be deployed by the operator as a software application (e.g., a Firewall VNFaaS in the NFV context)** **The firewall could also be deployed by the operator with some high capacity hardware in the data path controlled via SDN protocols.**
Transition Challenges	**Supporting both traditional and "XaaS-like" approaches adds additional operational complexity.** **The firewall service could be applied to all of the consumer's services – e.g., wireline as well as wireless. If the set of firewall policies is not the same for all of the consumer's services, that may introduce additional complexity.**
Advantages	**The availability of the SaaS-like service permits the consumer to subscribe to the firewall capability as needed rather than deploying capital; though perhaps a more important advantage is the administration of the firewall by the operator rather than the consumer may be more likely to result in better service quality through the Service Provider's more systematic operational procedures to maintain the firewall software at the latest release in order to deter emergent security threats.** **If the firewall is deployed by the operator as a software application (e.g., a Firewall VNFaaS in the NFV context), the resources for the firewall are likely to be consumed primarily during non-work hours when people are at home. This can result in optimal resource usage thus driving energy efficiency for the operators. This opens the door for new business models – e,g., usage based billing for firewalls.** **This approach can also provide a better customer experience through, for example, customized security profiles, dynamic-capacity scaling, and usage-based billing.**

4.5 Operator Programs Enterprise Infrastructure

4.5.1 Enterprise Exposes Network Function Virtualization Infrastructure as a Service (NFVIaaS) to Operator

Title	Enterprise exposes Network Function Virtualization Infrastructure as a Service (NFVIaaS) to Operator
Actors	Service Provider. Enterprise.
What is Programmable?	Both Service Provider and Enterprise have deployed a Network Function Virtualization Infrastructure (NFVI). The NFVI provides capabilities comparable to an IaaS cloud computing environment coupled with an NaaS that can provide on demand connectivity within some defined set of service termination points. The NFVI is a programmable infrastructure for authorized users.
Description	By agreement between the Enterprise and the Service Provider, the Service Provider is able to instantiate its own VNFs on the NFVI operated by the Enterprise. Using the VNFs running on both NFVIs, Service Provider can deliver end-end services across both NFVIs. Since VNFs are not all location independent this capability is expected to improve latency and reliability. It also enables compliance with some regulatory constraints on the location of processing and storage.
Traditional Approach	The traditional approach is a composite of manually, specially engineered wholesale services. Service Provider would need to negotiate individual leased lines, cloud computing resources, etc., with Enterprise. Assembling, and then operating the end-end service would be manually intensive and require significant coordination between the service providers – e.g., regarding troubleshooting and maintenance.
Programmability Approach	By agreement, Enterprise authorizes Service Provider to operate on NFVI that is operated by Enterprise. The NFVI provides the NaaS capabilities to dynamically connect to the authorized termination points within the NFVI. The NFVI provides the compute nodes with IaaS capabilities to provide the run time environment for the VNFs to be installed by Service Provider. Service Provider gets real-time updates on the status of the NFVI and is able to remedy malfunctions without the involvement of Enterprise.
Network Programmability	This IaaS environment is a generic computing platform programmable by authorized users. In this case, Service Provider is an authorized user. The NaaS provides on demand connectivity with authorized termination points within the NFVI. SDN based separation of data and control plane functions are expected to be an important enabler for scaling the NaaS capabilities.
Transition Challenges	Not all network functions can be economically virtualized and so the NFVI must interface to existing physical network elements. Deploying the NFVI impacts the existing OSS and BSSs which must be extended to support the VNF equivalents of existing Network Elements, as well as the orchestration of all these VNFs and the NFVI into an end to end service.

	Network operations based around physical connections become refocused on APIs and flows. Purpose built hardware becomes replaced by software defined VNFs. Manual deployments become replaced by automated orchestration of workloads. Sites with specialized NEs become replaced by cloud technology pods with generic computing infrastructure as the computer nodes of the NFVI. **Operator staff skill sets have to change to support a much more software intensive operations paradigm.** **Operator business practices have to support commercial offers of NFVIaaS as wholesale offers between operators with all the required details of service level agreements and other terms and conditions.**
Advantages	**The NFVI provides a shared infrastructure with pooled resources that can be dynamically allocated between different VNFs and other cloud computing workloads providing cost advantages over infrastructures dedicated to individual (siloed) services.** **NFVIaaS provides full control of the computing resource through administrative access to the VMs while trading the capital and operational cost of that computing infrastructure for a flexible, efficient rental of the capacity required. The NFVIaaS control of the infrastructure enables easier portability and interoperability of services developed by Service Provider in their own NFVI.** **The ability of the Service Provider to rapidly deploy services across the Enterprise's infrastructure should enable more rapid response to enterprise customer requests for service reconfigurations, etc.**

4.6 Enterprise Programs Enterprise Infrastructure

4.6.1 Bin Packing Data Center WAN Connectivity

Title	**Bin Packing data Center connectivity**[22]
Actors	**Enterprise Workload Orchestrator.** **Enterprise Data Center Applications.** **Enterprise Data Center Infrastructure.** **SP WAN Infrastructure.**
What is Programmable ?	**The Enterprise Data Center Infrastructure is programmable in response to the Enterprise Workload Orchestrator.**
Description	**The Enterprise Workload Orchestrator has some understanding of the communications demand profile expected when executing Enterprise Data Center Applications on the Enterprise Data Center Infrastructure.** **The Enterprise Workload Orchestrator schedules the Enterprise data center applications such that the peak communications load on the SP WAN Infrastructure is reduced.** **The Enterprise expenses SP WAN infrastructure that is dimensioned based on peak load expected. By reducing the peak load, the Enterprise can defer/reduce expenses associated with SP WAN services.**

[22] See< http://www.ietf.org/proceedings/84/slides/slides-84-sdnrg-4.pdf>.

	This orchestration activity is a form of traffic engineering. It resembles a bin packing algorithm as the orchestrator "packs" the communications demand profiles from the various Enterprise Data Center Applications to maintain the "bin" of available SP WAN capacity as full as possible, without exceeding the maximum capacity.
Traditional Approach	**The Enterprise Data Center applications are scheduled manually, or on demand with no consideration for the aggregate communications demand profile.** **Because such scheduling is not optimized for the communications demand profile, the aggregate communications demand may experience significant "peaks" as Enterprise Data Center applications coincide in attempting to use the SP WAN Infrastructure**
Programmability Approach	**Orchestrating the Enterprise Data Center Applications may require dynamic reconfiguration of the Enterprise Data Center Infrastructure, so that workloads can be rebalanced to run on different data centers, or on different portions of the Enterprise Data Center Infrastructure at different times.**
Network Programmability	**A programmatic interface exists between the Enterprise Workload Orchestrator and the Enterprise Data Center Infrastructure to permit dynamic reconfigurations of that Enterprise Data Center Infrastructure – e.g., via SDN.**
Transition Challenges	**The Enterprise Data Center Infrastructure must be capable of dynamic reconfiguration to support different Enterprise Data Center applications.** **SP WAN Infrastructure experiences increased utilization on interconnections between the Enterprise Data Centers.** **This is unlikely to impact Service Provider staffing, but may impact Enterprise IT staffing. The Enterprise may require some BSS support to provide the policies to support the orchestration activities.**
Advantages	**The Enterprise controls out of pocket expenses by controlling peak demand.** **The Enterprise provides better customer experience for its users by avoiding congestion issues associated with "peak" WAN communications.**

4.7 Consumer Programs Enterprise Infrastructure

A suitable use case of a Consumer programming Enterprise Infrastructure was not identified in the time available to develop this report. SaaS type services such as the use case identified in 4.4.1 could be extended to have the Consumer interacting with a SaaS functionality provided by an Enterprise, but such scenarios were not considered as identifying any additional requirements on infrastructure programmability beyond the existing use cases.

4.8 Operator Programs Consumer Infrastructure

4.8.1 Service Provider Configures CPE During Administrative Change

Title	Service Provider configures CPE during administrative change
Actors	Consumer. Service Provider.
What is Programmable?	The consumer has CPE devices (e.g., Smart TV, settop box, Internet of Things devices) that are configurable by the service provider.
Description	The consumer has CPE devices (e.g., Smart TV, settop box, Internet of Things devices) that are authorized and attached to a service provided by the service provider. These CPE devices are attached (e.g., through some form of tunnels) to specific infrastructure elements of the Service Provider). For administrative reasons (failures, maintenance, capacity upgrades, etc.), the Service Provider needs to rehome service to different infrastructure elements – e.g., authentication server. To effect the change in the CPE, the Service Provider pushes the appropriate policies down to the CPE through an SDN mechanism –e.g., open flow. The authorization to make such configuration changes would be included in the terms of service.
Traditional Approach	Traditional approaches might include truck rolls to replace CPE or manual remote login by Service Provider Staff, or e-mails from the service provider requesting customer actions. Where the service provider provides the CPE, existing mechanisms, like TR-069, can support re-homing.
Programmability Approach	This example is described in terms of configuration – e.g., changing destination addresses for particular services. In general, similar scenarios could be used to justify download of new software updates rather than simply configuration options.
Network Programmability	A protocol provides a mechanism to download new policies or software into the CPE.
Transition Challenges	Potentially large numbers of CPE devices may result in re-configuration storms after outage events.
Advantages	Better customer experience through avoiding service outage due to administrative operations. Control plane mechanisms (e.g., OpenFlow) provide a faster reconfiguration than typical management plane (e.g., SNMP) interactions. This may be important where the CPE configuration needs to be synchronized with some network change.

4.9 Enterprise Programs Consumer Infrastructure

4.9.1 Work Partition on Mobile Device

Title	Work Partition on Mobile Device[23]
Actors	Consumer. Enterprise. Service Provider.
What is Programmable?	The consumer has CPE mobile devices (e.g., Smart phone) that are configurable by the Service Provider to be partitioned into a personal and a work partition. The Work partition is programmable by the Enterprise.
Description	The consumer has CPE mobile devices (e.g., Smart Phone) These CPE devices are configurable by the Service Provider (though that is not essential for this use case). For administrative reasons (security, legal etc.), the Enterprise needs to program the Smart Phone. To effect the change in the CPE, the Enterprise causes an app to be downloaded to the device and executed. The authorization to make such configuration changes would be included in the terms of employment. This kind of business use of a personal device is sometimes referred to as “Bring Your Own Device” (BYOD).
Traditional Approach	Traditional approaches might have employees carrying separate devices for business and personal use.
Programmability Approach	This example is described in terms of download and execution of an app, but there could be other configuration changes also performed by the enterprise within the scope of the work partition on the device. For example, the - Enterprise may desire to delete any company email from a lost device.
Network Programmability	The CPE mobile device is the programmable entity. No network programmability is required for this use case. The Enterprise and Service Provider may well have back end operations running on cloud servers to provider automated operational support for large numbers of mobile devices.
Transition Challenges	Potentially large numbers of CPE devices may result in re-configuration storms after outage events.
Advantages	Better Enterprise Customer experience through better control of their data on CPE mobile devices. Better Consumer Customer experience through carrying only one device instead of two.

[23] See the AT&T Toggle application: <http://arstechnica.com/information-technology/2012/06/att-splits-phones-into-work-and-personal-partitions-on-any-carrier/>.

4.10 Consumer Programs Consumer Infrastructure

A suitable use case of a Consumer programming Consumer Infrastructure was not identified in the time available to develop this report.

5 OSS/BSS Impacts

Major issues in the OSS area include the need for abstraction in the following dimensions: (1) the configuration parameters of an infrastructure device (e.g., a network element) and (2) a model for how to describe the services. Much of the overhead for OSS comes from supporting a very disparate set of network device OS version/varieties and also supporting the different models which enable their services.

Network elements and the services they support often have different life cycles. The service definitions usually have a longer life-cycle than the underlying network elements. Meanwhile, the OSS/BSS systems are challenged to support the configuration/operation of both the service instances and the network elements at the same time.

Today's OSS sees the world from a fairly flat point of view. For example, it pushes new or edited configuration increments to a device and updates to a database within the OSS. With the use of tools like Netconf, there have been many enhancements in how OSS can automate the configuration actions. Many network devices also contain their own database for configuration and some provide for rollback and checking features. These enable a configuration to be sanity-checked before it is committed and also to return to a previous configuration version.

The concept of programmatic capabilities can enable an enhanced mechanism to create new and more dynamic services, which do not require a full OSS implementation. It also enables services which can be deployed using service models, rather than device-specific configuration templates. SDN offers a different approach, one which can help to address the complex problem of building an OSS stack for different devices and their services. SDN today has the concept of a controller, which is in effect a middleware layer that today can already perform certain functions; for example, configuring an Ethernet switch via specifications like OpenFlow. As SDN continues to develop, there is increased activity in the controller space around the Open Daylight (ODL) capabilities. Taking ODL as an example, this presents a model driven approach, where the network interfaces in the controller use common southbound interface.

The figure below shows a simplified representation of a model driven approach.

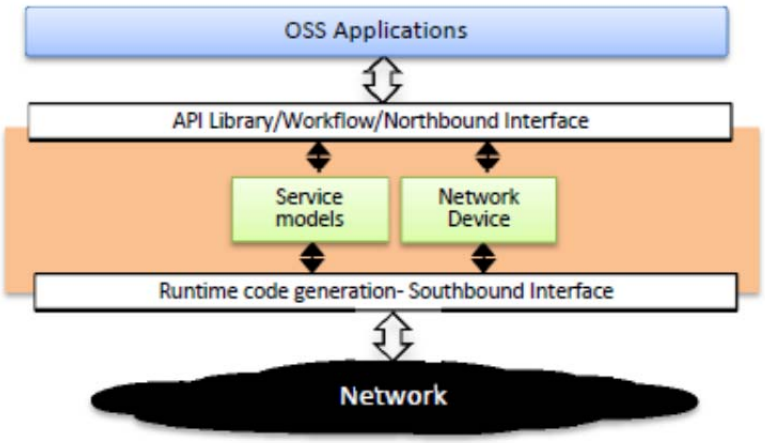

Figure 8: Model Driven Approach

The figure above illustrates two main types of models: the network device models and the service models. More model types can be built and exposed as required by further study.

These models and the engines for driving configuration are exposed using common APIs up to the Application layer and down to runtime/configuration functions where there is interaction with the network devices.

The Application layer is able to consume anything which is exposed and authorized. This avoids the classic vertical stack development models typically in use today.

One of the primary considerations for service providers is the business impact of changing or migrating a service that could be decades old. Often the service is designed to function around technology that is no longer supported or widely deployed, and with requirements that are long outdated. In addition there is often a Government Regulatory Framework around it, making it very difficult to change, e.g., ISDN. Each carrier will need to decide for itself if and when to migrate its services and OSS. Carriers must also decide whether to engage with their regulator, (e.g., FCC/OFCOM), to negotiate an end date for regulated services where the cost of migration is simply impractical.

It seems clear that today's operational environment does not continue to scale and support the rapid service creation environment. Future programmable operational environments should provide an abstraction layer modeling existing and future devices.

6 Staffing Skill Set Impacts

The transition to, and subsequent operation of, programmable networks is expected to broadly impact the functions of various Operator roles. Some roles will be more impacted than others. The following initial analysis lists the roles and impact expected:

- *Purchasing Agent* – New vendors, products, and services may mean that new relationships need to be formed, and new vendor evaluations performed.
- *Contract Management* - New vendors, products, and services may mean that new relationships need to be formed, and vendor metric tracking initiated.
- *Data Scientist* – New data collection, analytics processes, data analysis, insight development, and resulting actions may be needed.
- *Integration/Test Lab Engineers* – DevOp and other new methodologies to support agile execution in service development and delivery may be needed.
- *Network Management Engineers* – The movement from element management towards programming the network may require new programming skills.
- *OSS Support Engineers* – Integration among existing OSS and new orchestration components may require new skills.
- *HR/Management* – Staffing of new hires and retraining of existing personal in these roles may be needed.
- *Management/Staff Engineers* – Technical evaluation of new architectures, vendors, products, and services may be needed.
- *Business Architect* – New business capabilities may need new requirements for investment, workforce, information, technology, and processes.
- *Customer Service* – New services offered may require training and a shift to customer interactions related to those new services.

In order to achieve the skills transition, a mixture of internal and third party solutions (products and managed services) may be used for hiring, training, analytics, and network operations. Operators should develop a strategy to plan the balance over time between investment in internal skills-talent and out sourcing to a services model or managed services model.

6.1 New Employee Onboarding Use Case

Title	New employee onboarding use case
Actors	Service Provider.
What is Programmable?	The Service Provider's Infrastructure.
Description	The Service Provider has deployed a programmable infrastructure. The Service Provider hires a new employee to configure, deploy existing services, and design new services. The employee will need programming skills to maintain configuration control of the services deployed, tested, and developed.
Traditional Approach	Traditional approaches might have employees learning the custom capabilities of each individual network element in order to be able to configure those elements. The employee would also need to learn the specific configuration requirement for each specific service. The configurations would typically need to be manually installed and tested, though some automation may exist depending on the degree of OSS/BSS support with the service operator. Testing of new services is largely a manual process with significant possibilities of unforeseen service interactions. New service developments may require significant OSS/BSS developments further extending cycle times.
Programmability Approach	By having a common DevOps environment, the new employee can be productive across a variety of network services more quickly than in the traditional approach.
Network Programmability	The network programmability can be achieved through a variety of technologies such as SDN, NFV, OSS/BSS, etc. The important thing for this use case is the standardized operating environment in which the employee can develop, configure, test, deploy, etc.
Transition Challenges	Potentially large numbers of existing employees would need to be trained on the newly programmable operating environment.
Advantages	Better employee productivity for the service provider. Better job satisfaction from the employee because they have an immediate impact without being stuck in endless training.

6.2 Skills Pivots/Staffing Levels/HR Issues

The transition to programmable infrastructure creates a demand for staff that can design, develop, and program services on that infrastructure, as well as those who operate, administer, and maintain that infrastructure. While these programmable infrastructures are expected to support existing services, they are also expected to provide a platform enabling new services. The Enterprises and Consumers that use the services offered by these programmable infrastructures may thus be impacted by this increasing programmability as they take advantage of these new services. This transition may require training programs or other Human Resources actions to adjust the skill profile of the workforce.

The programmable infrastructure provides significant opportunities for more rapid deployment of services (e.g., at the click of a mouse – where infrastructure is available), and for automation of operational processes. This automation may require adjustments to staffing levels.

7 Financial/Tax Implications/Economic Modeling Considerations

7.1 Economic Advantages from Infrastructure Programmability

Adoption of NFV includes an expected[24] reduction in operational and capital costs. Figure 9 shows an example of NFV deployment on a central office basis. The current central office is assumed to have Network Elements (NEs) deployed in support of three services (A,B,C e.g., VPN, mobile services, and firewall), and an Element Management System – a type of Operations Support System concerned with the administration of NEs. For service reliability, the current central office deployments use simple redundancy (1:1 protection) of NEs – essentially duplicate NE deployments. The NEs are represented by the blue rectangles in Figure9, with the size of the rectangle intended to give some idea of the capacity of the NE, and the blue, green, and purple rectangles giving some indication of the current peak utilization of that capacity. For simplicity the NEs for services A, B, and C are illustrated to be of the same nominal capacity in the figure, but in general NEs supporting different services could have different capacities, and be of different types.

The future NFV-based central office is based on VNFs deployed as replacements for the NEs dedicated to each service. This use case assumes full virtualization of an existing NE into a corresponding VNF running in the NFVI. Similar deployment and operational concerns will exist for partially virtualized VNFs, but only the complete virtualization example is shown. In general, the capacity of compute domain nodes within the NFVI will not be the same as that of any given NE, and it will change as the technology implementing the compute domain nodes evolves over time, with the implementation efficiency of the VNFs and perhaps with many other factors. The compute domain nodes of Figure 9 are deployed in an n+m redundancy configuration. The VNFs in the future NFV-based central office are shown as the same blue, green, and purple colored rectangles as with the NEs; the colors indicating the service to which the VNF is applied. Failures of compute domain nodes become reductions on the maximum capacity available for the VNFs implementing services A, B, C. The orange rectangle illustrates the capacity required for the OSS/EMS functions in the current central office as well as the future NFVI-based central office. In this deployment example, the Service Provider achieves reductions in equipment costs and power consumption by replacing the stranded capacity from the redundancy scheme in the current central office with the shared spare capacity of the n+m redundancy scheme of the NFVI. Service A, B, C can be rapidly scaled through the allocation of additional capacity on the compute domain nodes to the VNFs, or by the deployment of additional VNF instances.

[24] See *Network Functions Virtualization – In*troductory White Paper http://portal.etsi.org/NFV/NFV_White_Paper.pdf.

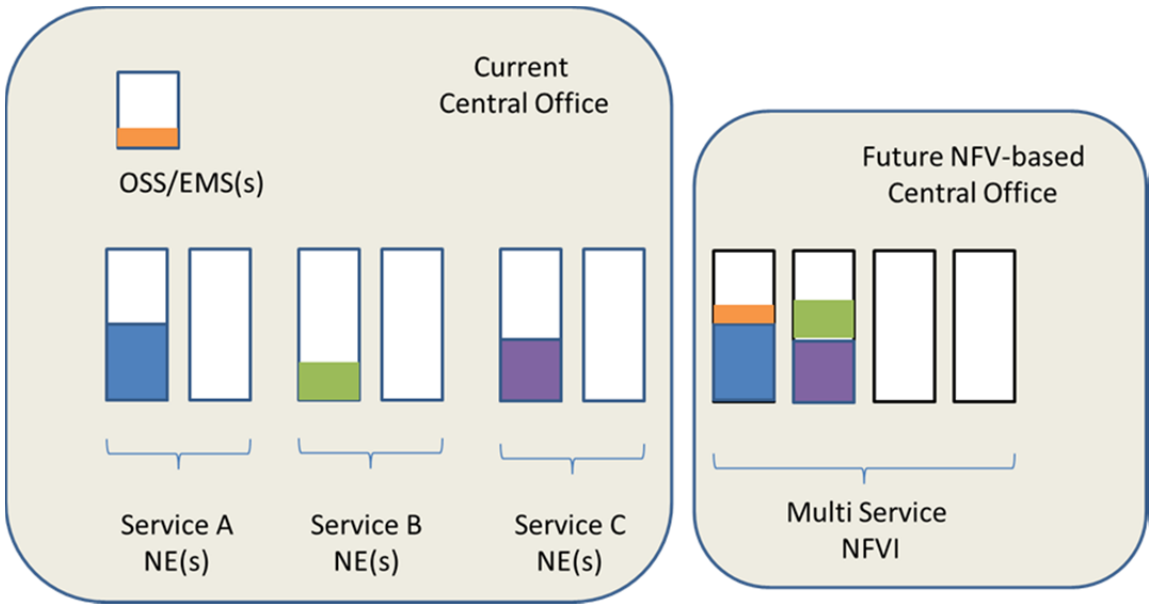

Figure 9 - Example NFV Deployment

The example of Figure 8 illustrates that a reduction in capital cost through elimination of stranded capacity may be feasible under certain circumstances due to aggregation. The reduction in the number of elements and the common NFVI operational environment may also have some impact on operational costs through staffing levels as well as reductions in nominal task time standards. In practice, actual deployment strategies may differ from this example for a variety of reasons.

7.2 Tax & Other Financial Considerations

The emphasis on programmability of the infrastructure implies a shift in the infrastructure to emphasize or expose software technologies. This may necessitate changes in business practices to recognize software licenses as separate infrastructure elements unbundled from the hardware platforms in which they execute.

Software licenses that are purchased as perpetual licenses may be considered capital expenses, but software license subscriptions would likely be considered operating expenses. The location of the software may also be important for state and local tax purposes.

Financing arrangements may include various security interests in assets. Commercial practices for security interests in physical assets are much more widely recognized than those for intangible assets like software licenses.

8 SDO Gap Analysis Summary

8.1 Open Source Projects

The availability of relevant open source software implementations can create de facto standards. Open source projects require more[25] than just the availability of source code. Some relevant open source projects include the following:

[25] <http://opensource.org/osd>

8.1.1 OpenStack/Neutron

OpenStack[26] is an open source project developing software for public and private clouds. Neutron[27] is an open source project within the OpenStack framework to provide Networking as a Service between interface devices managed by other OpenStack services.

8.1.2 OpenNaaS

OpenNaaS[28] is an open source project to develop a platform for Network as a Service resources including WAN devices such as IP routers and ROADMs.

8.1.3 OpenDaylight

Open Daylight[29] is a collaborative open source project organized under the Linux Foundation to foster innovation and create an open and transparent approach to software defined networking. This project is developing an open source SDN controller framework.

8.2 SDO Ecosystem

Organization	Scope of Work	Outlook of Work
BBF	Applying NFV in the broadband network.	Waiting for ISG stable work via liaison.
BBF Service Innovation & Market Requirements (SIMR) WG	SD-313: Business Requirements and Framework for SDN in Telecommunication Broadband Networks.	Internal document, not intended to be published. May influence SDN-LT work.
CableLabs SDN Study Group	Investigation of applicability of SDN techniques to Cable infrastructure.	Currently modeling CMTS as an OpenFlow switch.
CloudNFV Consortium	Develop an implementation expected to contain both commercial and open-source elements based on standard interfaces described by ETSI NFV specification.	July 2013 – end? The group's first project will be a cloud-based implementation of IP Multimedia Subsystem (IMS) infrastructure. Beta Demos in September 2013.

[26] <http://www.openstack.org/>

[27] <https://wiki.openstack.org/wiki/Neutron>

[28] <http://www.opennaas.org/>

[29] <http://www.opendaylight.org/>

DMTF	Virtualization Management (v 1.0 – ISO): managing Virtualization Hosts, provisioning and managing Virtual Machines, management of the Virtual Switches. Open Virtualization Format (OVF) (v1.1 - ISO): Interoperable Packaging format, migrate workloads from one vendor to another. Platform Management Volume 1 and 2 (ANSI): managing physical platforms, discovery, provisioning and configuration of platforms and platform components. Physical networking: L2, L3, DHCP, DNS. Virtual Networking: Edge Virtual Bridging. Storage Management Initiatives (SMI-S): By Storage Networking Industry Association (SNIA). System Management Architecture for Server Hardware (SMASH): Comprehensive management of all aspects of server hardware.	All the specs with exception of the Management Profiles on Network and Network Policy Management are published and available for use. Some are recognized as ISO standards. The Network and Network Policy management work is in progress (active development is done in the DMTF Network Services Management WG).
ETSI Network Functions Virtualization (NFV) Industry Specification Group (ISG)	Addressing hardware challenges by consolidating disparate network equipment types onto industry standard high volume servers, switches and storage, with important emphasis on implementing their functions in software. Specification requirements—not standards.	January 2013 start – five tracks: 1) Architecture of the Virtualization Infrastructure. 2) Software Architecture for Network Functions. 3) Performance. 4) Reliability and Availability. 5) Management and Orchestration.
IETF	Define relevant architectural frameworks and technical standards. Most Data Plane/Multi-tenancy work. Encourage an SDN ecosystem with tie-in to legacy protocols. No single working group specifically considering virtualization. Most relevant groups: PCE, FORCES, I2RS, NVO3, L2VPN, L3VPN. Also relevant: NetConf/XMPP, ALTO, HTTPbis.	Data Plane - Tunneling and multi-tenancy work early stages. Data & integration with legacy Control Plane (both DC and WAN) in early stages. Application Optimization work in early stages. Communication between application and network in very early stages.

Indiana University SDN Interoperability Lab (SDNLAB)	Encourages the development and adoption of standards-based Software-Defined Networking (SDN) technologies such as OpenFlow.	Developing testing tools, methodologies, and procedures. Supporting OpenFlow events and showcases. Contributing to Open Networking Foundation (ONF) working groups. Providing educational opportunities.
IRTF Software Define Networking Research Group (SDNRG)	The proposed charter focusses on hybrid SDN models, the SDNRG will provide objective definitions, metrics, and background research, with the goal of providing this information as input to protocol, network, and service design to SDOs and other standards producing organizations.	Open-ended; depends on future contributions.
ITU-T	No specific SG is assigned NFV work, although SG13 seems to take lead on work around NFV. SG13 WP6 will become initializing WG for NFV work (in conjunction with the main goal of SG13 - Future Networks). NVF work shall position ITU-T to define NFV and SDN for many other SDOs.	1) NFV Architecture. 2) NaaS Architecture (including NFV + SDN). 3) NaaS Connectivity • NW Function and Connectivity in flexible programmable way on L2/L3. • NaaS for any NW services (L4 and above). Work rest of 2013 and 2014.
ITU-T SG13 Q.14	Resolution 77 (WTSA) – Standardization work in ITU-T for software-defined networking. Y.FNsdn – Framework of software-defined networking for carrier networks in future networks (definition and overview, key properties, architecture, use cases). Y.FNsdn-fm – Requirements of formal specification and verification methods for SDN. NOTE: Virtualization is to be addressed through separate work items (Y.3011, Y.Fnvirtreq). February 2012 Q.21 (now Q.14) started to develop an SDN framework.	Y.FNsdn is ongoing. Y.FNsdn-fm is ongoing. Chartered to develop recommendations through 2016.
ITU-T SG11 Q.4, Q.6	Supplement on Framework of signalling for SDN. Signalling requirements for software-defined Broadband Access Networks. Scenarios and signalling requirements of unified intelligent programmable interfaces for IPv6.	Initiated as of February 2013.

Open Networking Foundation (ONF)	Define relevant architectural frameworks and technical standards. Encourage the deployment of an SDN ecosystem. Promote the SDN value proposition. Accelerate the adoption of SDN technologies and standards. Openflow stewardship.	Extend OpenFlow to address Optical Transport (L0-L1), along with upper layer protocols (L4-L7). Four initiatives: 1) Architecture and framework. 2) New transport. 3) Northbound API. 4) Forwarding abstractions.
TM-Forum	Understanding of relevancy and applicability of existing TM Forum standards (Frameworx, Management Interfaces, Metrics, and others) to the NFV domain, which in fact may be very well applicable, especially Information Framework, Management Interfaces, and the work in SLA Management area.	Developed a list of the candidate User Scenarios determine the priority. Organize work with other SDO of interest in the area of ETSI NFV. Determine applicability of existing TM Forum specs to NFV domain.

9 Other Considerations

The programming by another entity of the infrastructures of operators, enterprises, and consumers (see Figure 7) raises issues of authentication and authorization that rely on notions of identity associated with the particular service offer. It is a particular account that would be authorized to make use of a certain set of programmable services and resources and not just any such request. The programs should be robustly designed to accommodate the case that requested resources are not authorized. The consequences of unauthorized access should also be considered in the use cases. The use cases should be designed such that security failures by one party do not propagate to impact another party. The contractual terms and conditions associated with programming on another party's infrastructure should also identify and assign any liability consequences due to errors and omissions in the programming itself.

The development of a programming paradigm for infrastructure services has the possibility that some of those services may rely on cloud technologies for their implementation, with potentially similar risks in terms of security and privacy depending on the configuration of the particular infrastructure and service. Similarly, the transfer of information across organizational boundaries such as those in Figure 7 creates the potential for leakage of private information. Privacy in Cloud computing is an area of emerging technology concern[30].

The interfaces between the infrastructure entities of Figure 7 identify locations where communications services may be provided. The use cases in this report have not been formalized as service offers or classified within a regulatory framework. This report has not studied regulatory impacts on programmable networks.

[30] See ITU-T technology Watch Report, *Privacy in Cloud Computing*, March 2012.

10 Conclusions & Recommendations

While a detailed analysis of programmable network standards has not been done, it is clear that the work of initial leaders in SDN (ONF) and NFV (ETSI NFV ISG) is being extended by other organizations. It is also likely that much of today's information models developed by groups such as the TM Forum will likely be retained and extended as the OSS/BSS paradigm merges with service and control applications. Therefore, a comprehensive standardization program for programmable networks is not required.

Work is already underway in various industry organizations to define the overall architectural framework and control interfaces for SDN. However, no one is addressing the critical need for Northbound interfaces necessary to effectively manage a programmable network. Although consistent standards for northbound interfaces would be valuable, previous attempts to address this business problem have been unsuccessful. Leveraging open source frameworks such as Open Daylight, Open Stack could provide a way forward with an effective mechanism to link to vendor and service provider implementations.

While specifications often lead to interoperability for hardware technologies, for software technologies, the availability of open source implementations often leads to interoperable deployments. In some respects, Open Source is the more modern practice for industry standardization of software intensive functions. Work is already underway in various industry organizations to develop open source implementations using SDN and NFV concepts. Open source projects are typically developed within their own community governance arrangements and community notions of project scope. The scope of these projects has not been evaluated in comparison with the scale of industry requirements being unveiled in studies such as this from ATIS and others from ONF, ETSI, etc. A critical review of these open source projects in the context of the use cases identified above would provide a valuable service to the industry by identifying gaps between these projects and industry needs. These gaps may lead to expanded scope or activities in some of these projects, or to new open source projects being initiated.

Traditional inter-operator, narrowband provisioning has a well-defined, programmatic flow. An example of this is the Access Service Request (ASR), which provides long distance companies a method to order switched or special access from another company. One approach to develop a deeper understanding of the issues of programmability would be to develop APIs to support programmability of a particular network service. The NaaS concept from Figure 4 and the corresponding open source project OpenNaaS from 8.1.2 may provide a suitable basis for further exploration in the context of a specific technology – e.g., Ethernet. A similar mechanism is being defined for Ethernet services through collaboration between the Metro Ethernet Forum (MEF) and the ATIS Ordering and Billing Forum (OBF). SDN provides a similar function with the focus on real-time network control. Functionally, these interfaces can be considered a continuum of capabilities that include both real-time and non-real-time delivery. The practical recommendation is to analyze current NFV/SDN API development and the MEF/OBF provisioning activity for similarities in interface and data model. Identified similarities represent potential synergies and opportunities to accelerate both mechanisms. The MEF and ATIS/OBF are already developing some of the metadata to support ordering and billing of WAN Ethernet services.

As the southbound interfaces such as Openflow, I2RS are evolving and handled by SDOs like ONF, IETF there is value in continuing further study on the northbound APIs. A universal standard for northbound APIs may not be feasible – a pragmatic approach would be to promote standardized northbound interfaces in popular open source frameworks such as OpenDaylight or OpenStack. With the increasing popularity of open source SDN controllers and orchestration systems, this would benefit the vendor and operator community.

Deploying and operating such programmable infrastructures is an area where current industry practices and skill sets are likely to change from traditional approaches. The development of best practices to guide the industry through this transition would facilitate the ongoing, delivery of essential network services without interruptions.

Network programming languages and the software abstractions of networking functions are areas of technology emerging from research. Further study of these concepts may lead to greater adoption.

Open source orchestration systems such as Openstack independently orchestrate compute/storage resources and the network resources. This works well for Intra data center workloads where the network resources such as bandwidth are not a bottleneck. For distributed data centers interconnected by WAN, network resources, especially bandwidth, is a bottleneck. In this context, it is worth investigating joint orchestration of compute/storage/networking resources in an open source framework such as Openstack and the impact on northbound/southbound APIs.

Leveraging network intelligence is a new work item currently under discussion in the ATIS/TOPS council, Many of these aspects, especially real-time network analytics in the context of Layer 2/3/4 networking are relevant here. A good example is the Large Flow LAG load balancing use case in 4.2.1.2 above. It is recommended to draw synergies between the new work items in the areas of NFV/SDN and real-time network analytics.

The following topics are recommended for further analysis:

- Northbound APIs of programmable infrastructure, including:
    - Related open source initiatives.
    - Best current practices for agile operation of programmable infrastructure
    - Impacts on OSS/BSS evolution from exposure of APIs.
- Additional use cases for network analytics in the context of programmable infrastructure.
- Joint orchestration of computing/storage/networking aspects of programmable infrastructure.

Further analysis of these topics will require the following expertise from the service provider and supplier community:

- Lead network architects.
- Lead OSS architects.
- Network analytics expertise.
- NFV/cloud expertise.
- Open source/agile expertise.

Annex A: Acronyms

Acronym	Definition
BSS	Business Support System
DC	Data Center
FMO	Future Mode of Operation
IaaS	Infrastructure as a Service
LAG	Link Aggregation Group
M2M	Machine-to-machine
ODCA	Open Data Center Alliance
OSMINE	Operations Systems Modifications for the Integration of Network Elements
OSS	Operations Support System
PaaS	Platform as a Service
PMO	Present Mode of Operation
SDN	Software-Defined Networking
UC	Use Case

Annex B: Software Defined Networking Focus Group Members

Co-Chairs	
Steven Wright	AT&T
Ramki Krishnan	Brocade
Philip Jacobs	Cisco
Serge Manning	Huawei
Members	
Ken Biholar	Alcatel-Lucent
Kevin Sparks	Alcatel-Lucent
Cagatay Buyukkoc	AT&T
Michael Fargano	CenturyLink
Andrew McLachlan	Cisco
Dan Wilson	Ericsson
Shelby Kiewel	Ericsson
Stephen Hayes	Ericsson
Carroll Gray-Preston	GENBAND
David Foote	Hitachi
Joe Lenart	Hitachi
Marie-Paule Odini	HP
Vishwas Manral	HP
Jean-Yves Bernard	Rogers
David Holmes	Sprint
Mujahid Khan	Sprint
Ben Campbell	Tekelec
Eric McMurry	Tekelec