



White Paper

Service Orchestration & Network Virtualization: A Lifecycle View

Prepared by

Caroline Chappell
Principal Analyst, Cloud & NFV, Heavy Reading
www.heavyreading.com

on behalf of



www.cisco.com

December 2015

Introduction

Streamlining the design and delivery of new network services is a top priority for operators, as customers demand rapid service fulfillment. AT&T's announcement that it has cut provisioning times for Ethernet services by 95 percent, for example, has been a wake-up call to the market. Its competitors need to follow suit or risk losing business.

The enabling technology here is service orchestration – a new way of automating the lifecycle management of services in the network. But there are three aspects of service orchestration to bear in mind: It matters how lifecycle management is carried out; lifecycle management embraces service assurance as well as fulfillment; and service orchestration is responsible for end-to-end service delivery across the network, not just within a single physical or virtual network domain.

This paper argues that a service orchestration system that successfully streamlines network service delivery should be based on four pillars. First, service activation should be model-driven, based on precise and machine-readable data models that support automation without the need for code development. Second, and related to the first pillar, service lifecycle events (create, modify, delete) should be achieved by specifying, at a high level, the desired final state of a service, rather than by programming a detailed set of steps for each event. This "state convergence" approach, relying as it does on fast and reliable model-to-model mapping, is the secret sauce behind the rapid deployment of new services and responses to subsequent change requests on demand.

Two further pillars of service orchestration are the result of new operational requirements introduced by network virtualization. The third pillar is orchestrated assurance: If services are to be spun up, programmatically, on demand, their monitoring and assurance can't be an afterthought, handed off to a separate system. The service orchestration system needs to be responsible for ensuring that a service can be assured as soon as it is instantiated. Assurance becomes part of the service fulfillment process, driven by the same service models.

Finally, service orchestration needs to fulfil services end-to-end across the network, whether the devices it is configuring to support such services are physical or virtual, with traditional network management interfaces or under the command of an SDN controller and/or NFV management and orchestration (MANO) system. Service orchestration must be capable of programming both physical and network elements, since, for the foreseeable future, services will be implemented across multiple network domains consisting of both physical network functions (PNFs) and virtual network functions (VNFs).

The four pillars give rise to an extended definition of service orchestration that provides a basis for benchmarking the different approaches competing in this area. All four pillars contribute to the streamlined management of services so they can be delivered at the pace and with the cost efficiencies operators require today.

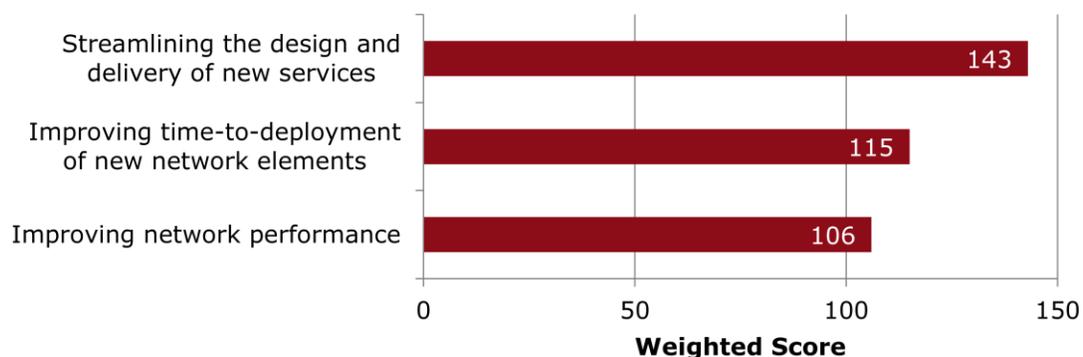
This white paper provides an overview of the four pillars underpinning service orchestration – support for data models, state convergence, orchestrated assurance and cross-domain support – then examines each pillar in detail.

Service Orchestration in the Virtualized Network

Streamlining Network Service Delivery Is a Top Priority

When Heavy Reading surveyed operators about their top three priorities for improving network operations, streamlining the design and delivery of new services was first by a significant margin.

Figure 1: Top Three Priorities for Improving Network Operations



Source: Heavy Reading (Score is a weighted calculation. Items ranked first are valued higher than the following ranks; the score is the sum of all weighted rank counts.)

The ability to provide network services on demand enables operators to provide a new level of customer experience. As AT&T pointed out in August 2015, it has cut provisioning times for Ethernet services by 95 percent and is receiving excellent feedback from customers as a result of programming the network using a service orchestration approach. Service orchestration is key to rapid service delivery, as we have explained in **Service Orchestration Requirements for a New Era of Networking**. And fast time to market is conferring significant competitive advantage on early adopters of service orchestration.

The Expanding Role of Service Orchestration

Service orchestration is an evolving term. Previously, we have defined it as **a new way of automating the activation of a service in the network, through the programmatic configuration of all elements participating in a service, in a single, end-to-end transaction.**

Service orchestration has primarily been associated with the deployment of customer-facing services in the network and the fulfilment of subsequent customer-driven change requests, including service retirement. But new operational requirements introduced by NFV are driving an expansion in the definition of service orchestration.

First, if services are to be spun up, programmatically, on demand, their monitoring and assurance can't be an afterthought. There should be no delay while the assurance aspects of a service are ingested and mapped to a new management model by a separate system. The service orchestration system needs to be responsible for ensuring that a service can be assured as soon as it is instantiated. Assurance becomes part of the service fulfilment process, driven by the same service models.

Second, as the network virtualizes, service orchestration must also be capable of programming both physical and network elements, since, for the foreseeable future, services will be implemented across a hybrid network topology consisting of both PNFs and VNFs.

These two requirements – the ability to span physical and virtual network domains and support for service assurance at the same time as service activation – are extending the definition of service orchestration. We now define service orchestration as **a new way of automating the lifecycle management of services in the network, through the programmatic configuration of all physical and virtual elements participating in a service, in a single, end-to-end transaction and the automated monitoring and redeployment of services to maintain customer SLAs.**

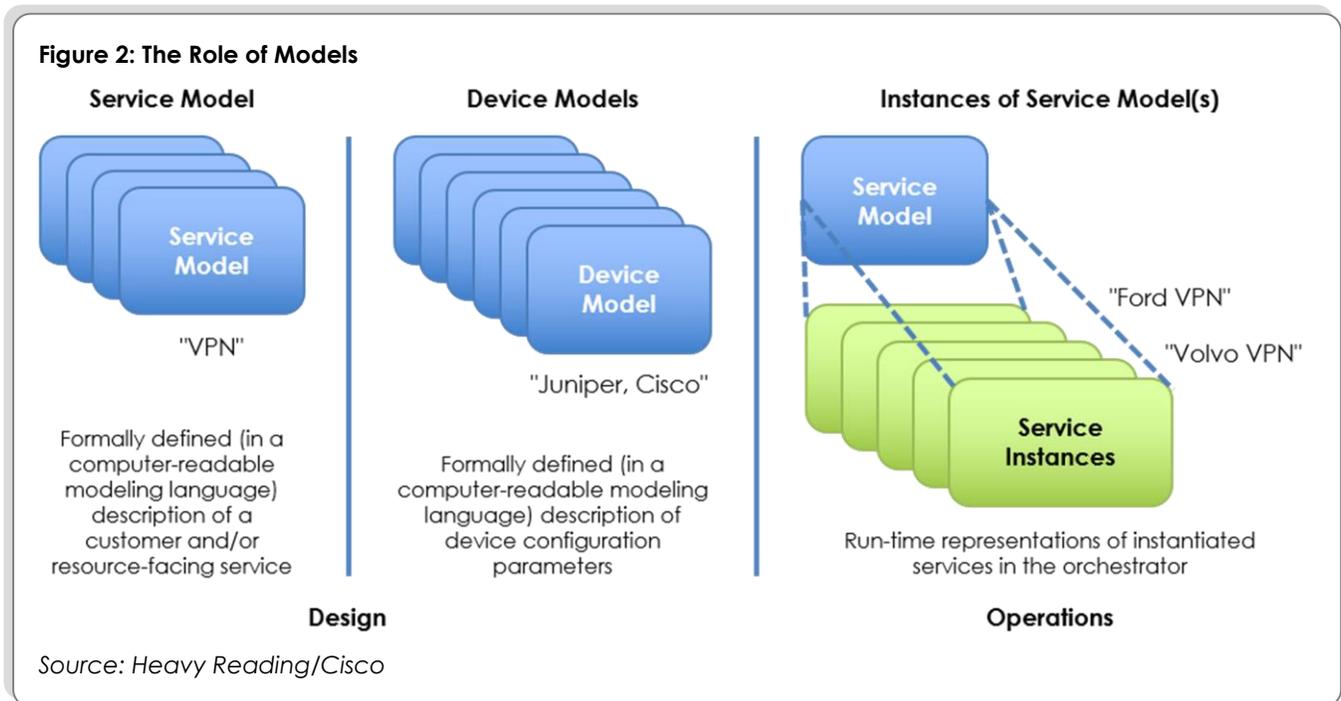
Four Pillars of Service Orchestration

A service orchestration system that meets this definition is therefore based on the following four pillars:

- **Data models and data model mapping** as a basis for the automation of configuration tasks. Services need to be modeled in a precise and semantically rich way using a standardized data modeling language that can be manipulated programmatically. If devices are modeled in the same language, it is possible to map service models to device models declaratively, without coding. This supports the rapid automation of service delivery.
- **A declarative approach to service lifecycle management (create, modify, delete) using state convergence.** State convergence works with data models to minimize overhead and increase reliability in activating services and changing them post-deployment, cutting service delivery cost and time.
- **Service monitoring and SLA management (assurance).** Service orchestration is becoming key to ensuring that services can be assured synchronously with activation and in an efficient manner. To do this, the service orchestration system will need to instantiate key performance indicators (KPIs) defined for each service model at runtime and use the orchestration system's real-time, stateful view of the network as well as input from service monitoring devices to detect impacts on those KPIs. The service orchestration can also respond to remediation decisions by redeploying services to restore customers' SLAs.
- **Programmatic configuration of services across physical and virtual network domains.** Service orchestration needs to fulfil services end-to-end across the network, whether the devices it is configuring to support such services are physical or virtual, with traditional network management interfaces or under the command of an SDN controller. The NFV MANO stack also uses the term "service orchestration," but in a MANO context, this has a specific meaning. It refers to the instantiation of a virtualized "network service," an application which may eventually be configured as a component of an end-to-end service. As network elements supporting end-to-end services are virtualized, an end-to-end service orchestration system will rely on the MANO to carry out the subset of orchestration functionality that applies to VNFs. For example, service orchestration should trigger the MANO to launch, configure and monitor the VNFs and complex service chains of VNFs which substitute for PNFs in virtualizing networks. End-to-end service orchestration by definition needs to work with different network management domains, including the MANO and SDN controllers.

Pillar 1: Data Models & Data Model Mapping

As we have explained in previous white papers, for example, **Service Orchestration Requirements for a New Era of Networking**, the inputs to a service orchestration system are data models that formally describe the services that need to be orchestrated and the devices that need to be configured (see **Figure 2**). Separating data models from the orchestration system is key: The service orchestrator needs to remain "stupid" with regard to domain-specific service knowledge so that it can orchestrate any service model given to it.



The data modeling language should be semantically rich, with a grammar that can be programmatically manipulated ("parsed") and instantiated automatically by the service orchestration system. Modeling services in such a language eliminates problems with the current method of describing services using natural language in Word documents or Unified Modeling Language (UML) diagrams. Natural-language service descriptions need to be interpreted by programmers and turned into workflows that instantiate the service for a specific customer across the CLIs or APIs of network devices. This is typically an expensive, time-consuming and potentially never-ending professional services engagement, as new workflows need to be created for every service instance and for each new operation that needs to be carried out on the service and set of devices used to implement it.

The data modeling language should be standardized and used to model network devices, both physical and virtual, as well as services. The service orchestration system is then able to map service models onto device models directly, without an intervening, proprietary data transformation step. This supports rapid, reliable and standards-based automation of the network configuration process. It also provides operators with the ability to reuse the same service models against devices from different vendors, for example, in different network locations, and/or interchangeably use physical or virtual network elements.

Pillar 2: State Convergence

Traditional Approaches to Automating Service Provisioning

Rapid service deployment, modification and deletion are critical aspects of service agility, enabling operators to respond dynamically to customer demand. Operators need a fast and efficient way of defining the operational steps needed to launch, change or tear down customer-facing services across their networks and a means of automating these steps for rapid and error-free execution.

Fulfilling service requests in today's complex, multi-vendor network environment is becoming more complicated as a result of NFV and the virtualized platforms and elements that must be started and managed alongside traditional physical network appliances. Ideally, operators want a **declarative**, or **intent-based**, way of defining service deployment, modification or deletion events without reference to the specific network elements affected. In other words, they want to describe, very simply and at a high level of abstraction, the desired final state of the service without having to go into detail about the steps needed to achieve it. This enables the operational aspects of new services to be described quickly and with a high degree of flexibility, since the operational description is abstracted away from implementation details and the latter are handled by the service orchestration system at runtime.

At present, there are two widely-used ways of defining how service components should be deployed, changed or deleted in response to customer service orders. Both, however, are **imperative**, which means they require a detailed level of programming to achieve service lifecycle management goals. As a result, neither satisfactorily meets the service agility ideal.

Workflow-based service lifecycle management definition: Provisioning steps for any given customer-facing service are defined procedurally as a series of workflows. On the plus side, workflow-based service definitions are easy for humans to understand and suit service situations where there are a high number of manual tasks, requiring manual approvals and truck rolls, for example. The disadvantage of a workflow-based approach is limited abstraction: service lifecycle definitions are inextricably linked with implementation detail.

This detail can be encapsulated in reusable templates but nevertheless, changes to the network at implementation level will require changes to workflows/templates. Creating and maintaining the set of deployment/modification/deletion workflows/templates that tell a particular service how to achieve its final, provisioned state, is a never-ending task, since workflows need to be scripted for every possible network change. Nor can a set of lifecycle management operations, expressed as workflows, be automatically ported across platforms and network elements. Workflow-based service provisioning does not satisfy the requirement for a simple, declarative way of launching and managing a service.

State machine-based service lifecycle management definition: The set of operations associated with service lifecycle management can be defined using a state machine approach. A state machine models the complete set of actions that need to be taken to deploy, change or delete the service and does preserve abstraction between service model and implementation detail.

Both workflow and state machine approaches can coexist and are often used together, with workflow engines handing off specific lifecycle management tasks to

a state machine. However, though a less messy and open-ended approach than scripting hundreds of workflow statements, state machines are notoriously difficult to design. In complex service situations a very high level of skill is needed to model all the possible ways in which the services can achieve a final, fulfilled state. Given the speed and low cost with which operators want to be able to deploy and change future services in their networks, the state machine approach is unlikely to prove satisfactory for service management.

To summarize, workflows tell a provisioning system, in detail, which steps to take to reach a final, fulfilled state, with all of a service's components up and running and properly configured. The state machine approach models in advance how the service is to reach that final state. But both approaches have the drawback that the service designer needs to define all possible workflows/state-transitions for all possible service changes and lifecycle events – a difficult, time-consuming and expensive task.

The Concept of State Convergence

Instead of modeling the entire final deployed state of a service in the network in a state machine, a **state convergence approach** models a deployment as a **mapping** from service input parameters to desired network state. The orchestrator should then be able to converge toward that final desired network state, deriving the steps it needs to take from the service input parameters it has been given.

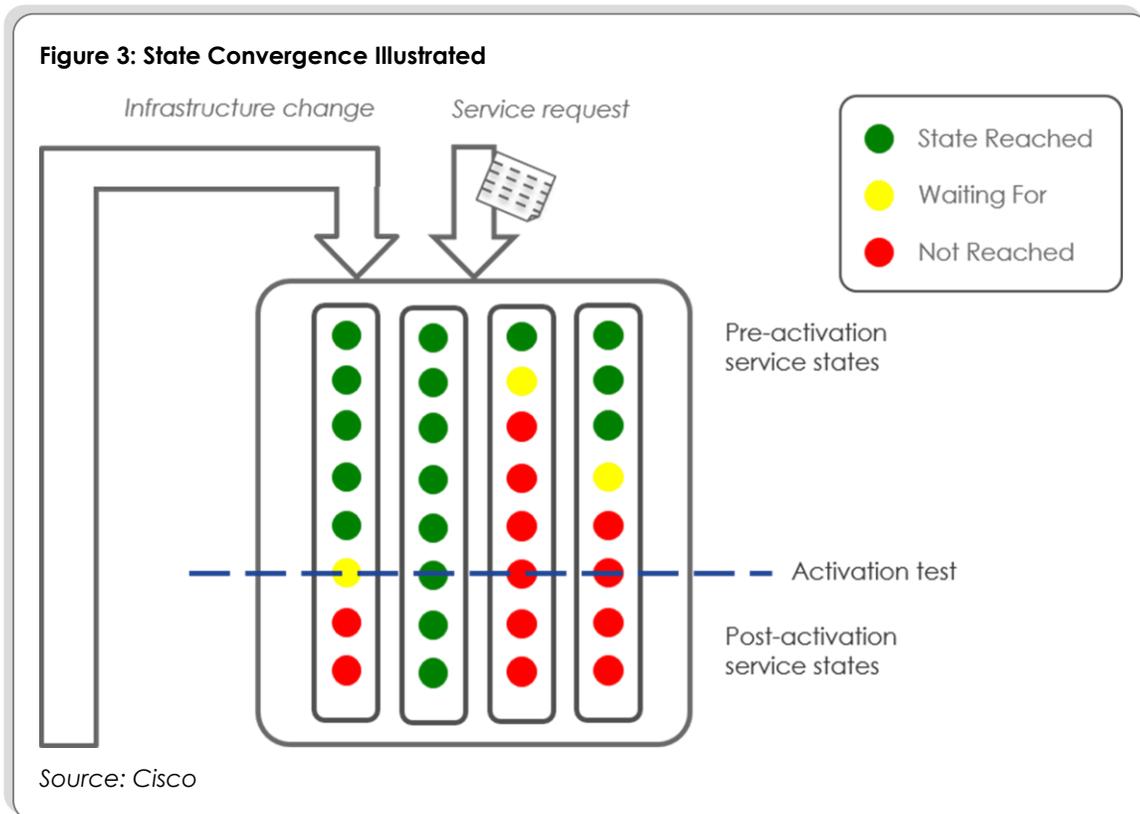
The orchestration system converges at runtime on the desired state of a service by re-running deployment steps as many times as is needed until the network has reached the final state as specified by the service input parameters.

In a **normal case**, there are three possible orchestration scenarios:

- **Case 1:** The deployment steps simply run straight through as all resources needed are available and supplied.
- **Case 2:** The orchestration system has to trigger external systems to provide resources – for example, virtual resources or an IP address, and the deployment is temporarily paused until these requests have been fulfilled. In this scenario, network changes that have already occurred as part of the deployment transaction remain in place. When the orchestration system is notified, for example, that the required virtual resource or IP address has been supplied, it starts the deployment again from the top. In this way, it checks that all previous network changes are still valid. However, to avoid the overhead of unnecessary changes to the network, an orchestration system taking this approach must be able to recognize steps that have already been completed and skip over them to the next uncompleted step. It does this by calculating from the service input parameters the minimum number of changes it needs to carry out to go from the current network state to the final network state. It needs this "minimum diff" calculation capability for all service lifecycle operations (e.g., create, modify, delete). Operators can be confident that re-running the deployment as many times as needed to converge on the final deployed state will have no impact on completed steps.
- **Case 3:** If, while the deployment has been paused at a certain step, a previous step experiences a failure – for example, the orchestration system detects that a virtual machine goes down – it must be able to go backward and fix the previous step before proceeding. The deployment is complete when it runs as Case 1 and no further network changes are needed.

The possibility of **error cases** demands that state convergence must be transactional. If the orchestration system has applied configurations to the network but cannot converge on a final state for some reason – for example, in an NFV scenario, because a VNF has failed and cannot be restored – it needs to be able to roll back those configurations to the start of the transaction. The orchestrator must be able to "un-deploy" the service steps, in the correct sequence, to preserve network and operational consistency.

A service orchestration system that supports state convergence will need to be able to trigger state-changes to other systems, such as the IP address system to get the service an IP address, or OpenStack to spin up a virtual function for the service. The service orchestration system will also require an event-listening mechanism, based on a "publish and subscribe" messaging model, so that it can detect responses to its request(s) and re-run the relevant service deployment workflow(s) from the virtual infrastructure or external systems.



Pillar 3: Service Orchestration & Assurance

Challenges With Traditional Service Assurance

Services are deployed on behalf of customers so the customer's view of service performance is of paramount importance to the operator's business. The goal of service assurance is to ensure that customers receive the service experience they have contracted for by monitoring service KPIs and remediating any problems that affect them. However, the three most relevant KPIs from an end-user perspective for one service type, such as latency, jitter and delay, are not necessarily the same for another, which may be more concerned with buffer size and play rate.

When operators want to deliver services on demand, those services have to be instantiated as "assurable entities," ready from the start to be monitored and assured. This is very difficult to achieve with a traditional service assurance approach. Traditional service assurance systems work from the bottom up to "discover" a service and its KPIs by sifting and correlating a huge amount of device data, both real-time data from the network itself and offline data sitting in inventory systems. However, creating effective algorithms that can successfully deduce service performance from huge numbers of device KPIs has proven extremely difficult over the years.

This challenge is compounded by unsatisfactory data quality in many inventory systems and the dynamic nature of NFV, which makes VNF-to-service mapping and consequent bottom-up calculation of service KPIs very hard. Instead of fixed, physical devices, services in a virtualized network may run across arbitrary numbers of VNFs that can scale up or down at any point in time.

Traditional service assurance systems also work after the fact of service fulfilment. They are rarely synchronized with the service provisioning process, nor do they work from the same set of service models. This means that where services are delivered on demand, there may be an unacceptable time lag between the activation and assurance of a service from a user point of view.

Orchestrating Assurance: A New Approach

Orchestrating assurance as part of the service fulfilment process addresses the shortcomings of traditional systems. However, it adds four new requirements to the way in which operators handle service orchestration. Operators will need to:

- **Define service KPIs** as part of the service models they give to the service orchestration system. Service KPIs need to be reflected in YANG service models, for example, alongside configuration attributes. In fact, operators will need to put equal effort into modeling SLA thresholds and configuration parameters for each service since service KPIs closely reflect the type of service that is being monitored. The number of KPIs need not be extensive, but they do need to be relevant, not generic, in contrast to general-purpose service assurance systems.
- **Measure service KPIs directly at true service end points** from the user perspective so that the service orchestration system has accurate, real-time knowledge of the customer's service experience. Measurement implies the use of monitoring functions in end devices. Virtual probes, customized to understand the KPIs for a specific service type and deployed to an end point as part of a service orchestration, are ideal candidates for this task.

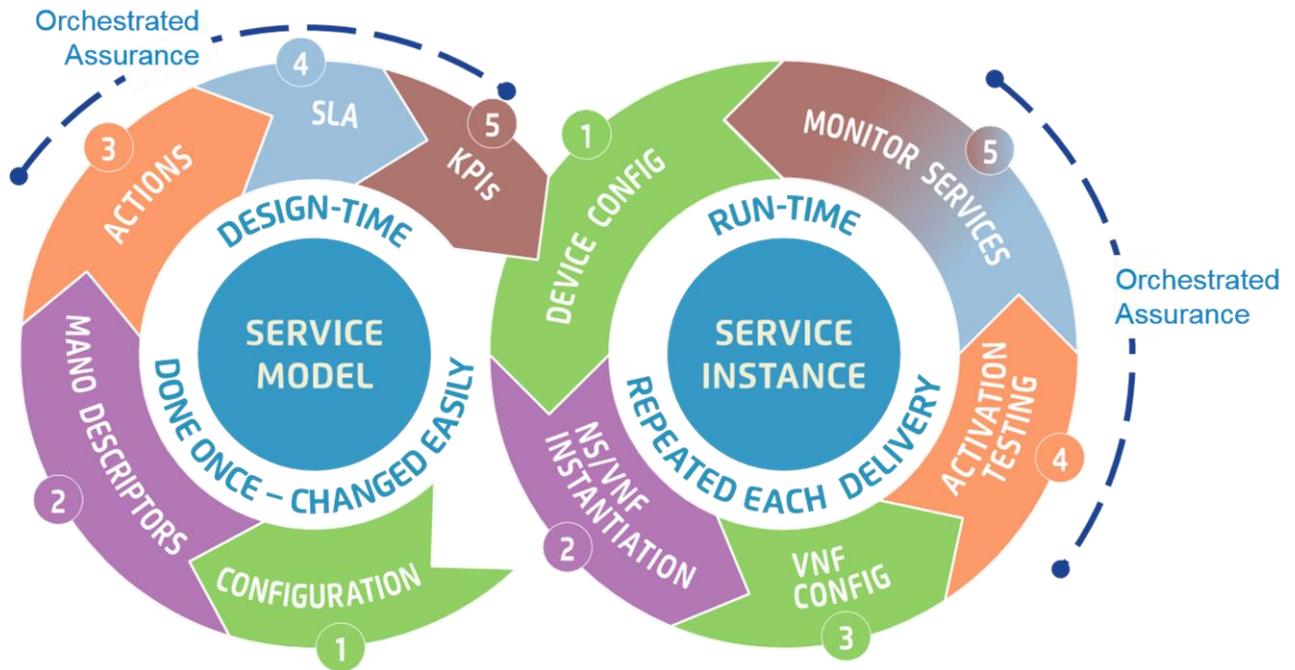
Virtual probes are inexpensive and can therefore be placed flexibly at the network edge by the service orchestration system, in conjunction with an NFV MANO system, on a per-customer, per-service instance basis. Probe data populates the KPIs regardless of how or where the service is implemented in the network at any given time.

- **Run activation tests that prove a service instance delivers on its KPIs** as the final stage of the provisioning process. This is a best practice recommendation that originated with the Metro Ethernet Forum (MEF) but which should become the norm for delivering any customer-facing service. An activation test is a stress test of the service that proves the service has been provisioned correctly and that it works. Many operators currently do not provide "birth certificates" for their services, based on activation tests, although by doing so, they enhance the customer experience, reduce the likelihood of service failure and give themselves a competitive advantage. Activation tests take service orchestration into DevOps territory, as in the DevOps world, developers must design tests before they develop application code.
- **Ensure that the service is "assurable" from the moment of instantiation.** The service orchestration system should inform external assurance systems, responsible for root cause analysis and remediation, about the service and trigger SLA monitoring as part of a service deployment script. It should similarly request removal of service data from those systems when it tears a service down.
- **Automate orchestration actions in response to SLA violations.** By measuring customer service experience at the true end-point, if there is a fault somewhere in the network that is affecting service KPIs, operators know immediately which customer(s) and service(s) are affected without waiting for this information to percolate up through layers of physical and virtual network devices. In around 50 percent of cases, service degradations are caused by configuration errors or non-optimal configurations that the service orchestration system, as master of configuration data, can detect and rectify. In the virtualized network, the service orchestrator can use its state convergence approach to trigger the reconfiguration and redeployment of VNFs to stabilize service performance. If the underlying NFV infrastructure is at fault – virtual machines are failing or moving, for example – this can be flagged as an event to the service orchestrator, which again, uses state convergence to redeploy the service in a closed feedback loop. Faults in the physical network will need troubleshooting and remediation by external assurance systems to identify affected devices.

The expansion of service orchestration to address both fulfilment and assurance aspects of the service lifecycle will be mandatory in the NFV world, but it is highly desirable in the traditional physical network, too. Addressing service assurance from a customer and service perspective supports the agility and flexibility that operators are seeking by preserving the principle of abstracting service models from their implementation in devices.

This separation of concerns is critical for NFV, where services may be deployed and redeployed throughout their lifetime on multiple instances of the same VNF. NFV also enables closed loop feedback between fulfilment and assurance, where a state convergence-driven service orchestration can ensure a service is assurable as soon as it is activated and, triggered by SLA violations, can automatically redeploy affected parts of a service.

Figure 4: Orchestrated Assurance



Source: Cisco, Netrounds

Pillar 4: Cross-Domain Service Delivery

Service Orchestration & the NFV MANO

Operators tell Heavy Reading that they want a single service orchestration system capable of fulfilling customer-facing services end-to-end across the network, regardless of whether network elements are physical or have been virtualized and despite the means used to interface with those elements – an SDN controller, NFV MANO or traditional CLI. They recognize that the network is likely to consist of multiple domains, physical and virtualized, for the foreseeable future.

A service orchestration layer that sits above the NFV MANO is the best repository for end-to-end customer-facing service models in a hybrid network. By itself, the NFV MANO only addresses the activation and assurance of VNFs and service chains of VNFs. It cannot manage end-to-end customer-facing services deployed using a combination of VNFs and PNFs.

When deploying customer-facing services across a hybrid network, the higher level service orchestration system will need to query the NFV MANO to see if the required VNFs/network services are available, trigger it to instantiate VNFs/network service(s) if not, and receive notification from the NFV MANO of VNF or NFV infrastructure (NFVI) performance-impacting events that require the service orchestration system to redeploy the service. In a hybrid scenario, the service orchestration system makes use of the NFV MANO's two separate but interconnected areas of functionality:

- Management and orchestration of NFVI and the virtualized compute, storage and network resources from which the NFVI is composed in order to provide appropriate execution environments for VNFs. In the current ETSI NFV architecture description, this functionality is divided between the Virtual Infrastructure Manager (VIM) and the NFV Orchestrator (NFV-O).
- Management of the application lifecycle of VNFs and complex chains of VNFs that comprise a single application, such as the virtual Evolved Packet Core (vEPC). In the current ETSI NFV architecture description, this functionality is divided between the VNF Manager (VNFM) and NFV-O. The VNFM manages simple, atomic VNFs, while the NFV-O manages VNFs that are chained together into what ETSI NFV confusingly calls "network services," but which can still be treated as applications from a cloud point of view.

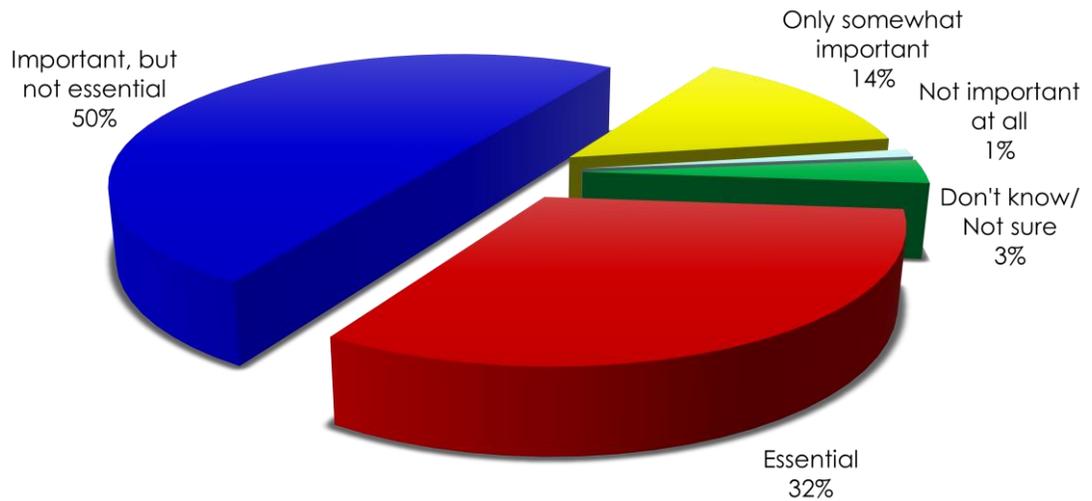
As we pointed out in **Deploying Virtual Network Functions: the Complementary Role of NETCONF/YANG**, the VNFM and NFV-O within the NFV MANO can use pre-defined templates to manage the cloud aspects of VNF/network service lifecycles. This paper refers to both VNFs and network services as "NFV applications," since both can be treated similarly. These cloud lifecycle management operations consist of:

- Install/uninstall/reinstall an application instance
- Start an application instance, or restart if it crashes or needs to be upgraded with new features/configurations
- Carry out initial "Day Zero" (operational readiness) configuration of the application instance

Customer-specific configuration and application-level (rather than infrastructure-level) monitoring are still the domain of each VNF's element management system (EMS). EMSs currently talk to OSS, of which the service orchestration system is a part, not to the NFV MANO.

VNFs in this landscape are likely to be multi-tenant at present – that is, they will support multiple customer services, just like their PNF counterparts. So although the NFV MANO creates VNFs/network services with an "operationally ready" configuration, they will need further configuration, by the service orchestration system, to support customer-specific service instances. Many operators would like VNFs to share the same EMS as their physical counterparts (see **Figure 5**) to ensure that network element configuration is operationally consistent, regardless of how the element is implemented.

Figure 5: The Importance of a Common Element Manager

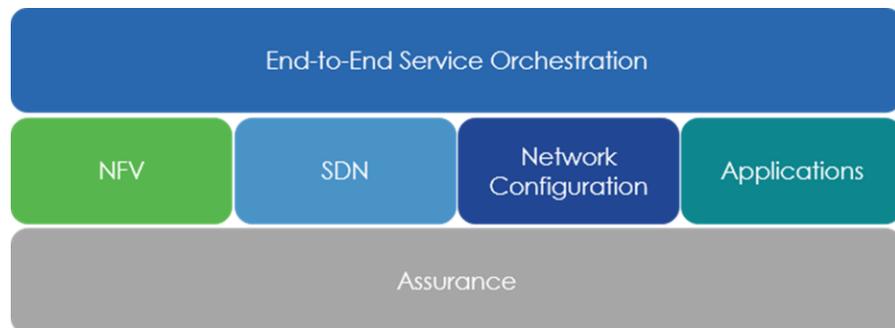


Source: Heavy Reading NFV MANO and OSS survey 2015

Service Orchestration & the SDN Controller

In an SDN scenario, the service orchestration layer holds the end-to-end service models but triggers the SDN controller to carry out the activation of PNFs and VNFs. **Figure 6** is a simplistic view of how service orchestration spans traditional physical, virtual and SDN-based network domains.

Figure 6: Orchestration Spans Multiple Domains



Source: Cisco

Conclusion

As service orchestration is increasingly applied to the activation of services in the virtualized as well as the physical network, its scope is expanding.

To carry out service activation cost-effectively and provide the service agility that operators want, service orchestration needs to be model-driven. That is, instead of activation based on the never-ending task of workflow definition on a per-service basis, service orchestration should use the concept of state convergence, which requires services and network devices to be modeled in a standardized modeling language by network domain experts. These models can then be declaratively mapped to one another without the need for coding and in a highly flexible way.

Support for data models and state convergence represent the first two pillars of service orchestration. However, as the network is virtualized, it is introducing new domains – the NFV MANO and SDN controllers – that need to be supported when a service is deployed end-to-end across both physical and virtualized network elements. And as services can be deployed more quickly in all types of network – physical, hybrid and virtual – thanks to model-driven automation, they need to be assured as soon as they are activated, or the benefits of service agility are lost.

A service orchestration system should thus encompass two further pillars: the ability to orchestrate across both physical and virtual networks, whether devices are configured via traditional (CLI/API) means, the NFV MANO or an SDN controller; and support for orchestrated assurance – i.e., the ability to assure both that the activated service has been properly deployed and that it is delivering on its SLA at runtime.

Leading operators are already transforming service delivery times and customer experience through the use of service orchestration based on these four pillars. Those not yet taking advantage of service orchestration should carefully evaluate competing systems to ensure they support all four pillars for a future-proof approach to network service lifecycle management.

About Cisco

Cisco (Nasdaq: CSCO) is the worldwide leader in IT that helps companies seize the opportunities of tomorrow by proving that amazing things can happen when you connect the previously unconnected. For ongoing news, please go to newsroom.cisco.com. Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. A listing of Cisco's trademarks can be found at www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company.