There is described a method for providing a Virtualized Network Function (VNF) according to Network Service (NS) requirements. The method comprises selecting an on-boarded VNF descriptor (VNFD) from a VNF catalogue, configuring parameters of the selected on-boarded VNFD according to the requirements of the NS and instantiating a VNF according to the configured on-boarded VNFD. There is also described a method for providing a Network Service (NS). The method comprises selecting an on-boarded NS Descriptor (NSD) from an NS catalogue, modifying NSD information of the selected on-boarded NSD and instantiating the NS according to the modified on-boarded NSD.

- Selecting an on-boarded NS Descriptor (NSD) from a NS catalogue 901
- Modifying NSD information of the selected on-boarded NSD 902
- Instantiating the NS according to the modified on-boarded NSD 903
- Creating a NS Record (NSR) 904
- Adding Network Function Virtualization Infrastructure (NFVI) resources reserved for an NS instance of the NS, an NS status, and a logging capability for audit purpose to the NSR 905
receive a request for a new network service (NS) 501

on-board the NS 502

initiate VNFs configuration generation by sending a configuration generation request to at least one VNFM 503

receive VNFRs created by the VNFM 504

generate the NS configuration 505

initiate deployment of the NS 506

checking for resource availability for each VNF 507

initiating reservation of resource for the VNFs 508

sending reservation acknowledgment to corresponding VNFM 509

sending VIM identifier to inform VNFM where to deploy each VNF, thereby triggering deployment of the VNFs by the VNFM 510

complete deployment of the NS 511

send a request the VIM for network connectivity creation 512

receive a confirmation from the VIM of successful creation of network connectivity 513

request the VIM to connect VNFs to a network 514

Figure 5
providing default values for any missing information elements in the VNFD as well as applying policies and constraints of the given NFV system 601

selecting VNFs from the VNFCatalogue 602

tailoring of VNFDs, corresponding to the selected VNFs, for a specific NS based on the NSReq from the tenant where various VNF/VNFC deployment constraints and virtualized network resource requirements might be added/updated in the VNFD by the OSS/BSS 603

the VNFM/EM configuring VNF-specific data for VNF instantiation 604

creating the VNFR 605

performing the VDU-level configuration 606

Figure 6

providing with the default values for any missing information elements in the NSD as well as applying policies of the given NFV system 701

instantiating the NS by creating an NSR by adding additional information to the on-boarded NSD, such as information on NFVI resources reserved for this NS instance, the NS status and the logging capabilities 702

Figure 7
**Figure 8**

1. selecting an on-boarded VNF descriptor (VNFD) from a VNF catalogue
2. configuring parameters of the selected on-boarded VNFD according to the requirements of the NS
3. instantiating a VNF according to the configured on-boarded VNFD
4. creating a VNF Record (VNFR)

**Figure 9**

1. selecting an on-boarded NS Descriptor (NSD) from a NS catalogue
2. modifying NSD information of the selected on-boarded NSD
3. instantiating the NS according to the modified on-boarded NSD
4. creating a NS Record (NSR)
5. adding Network Function Virtualization Infrastructure (NFVI) resources reserved for an NS instance of the NS, an NS status, and a logging capability for audit purpose to the NSR
A NETWORK SERVICE DESIGN AND DEPLOYMENT PROCESS FOR NFV SYSTEMS

TECHNICAL FIELD

[0001] The present disclosure relates to network service design and deployment.

BACKGROUND

[0002] Network Functions Virtualization (NFV) is an emerging paradigm that builds on cloud computing and the virtualization technology to eliminate the drawbacks of traditional physical network infrastructure and that enables rapid provisioning of network services (NSs). With NFV, there is no need for the deployment of a wide range of network equipment, which requires substantial capital and operating expenses. NFV exploits the virtualization technology and cloud resources and remodels the physical devices into virtual entities known as Virtualized Network Functions (VNFs) implemented as software packages.

[0003] The NFV reference architectural framework, standardized by the European Telecommunications Standards Institute (ETSI) and adopted by Topology and Orchestration Specification for Cloud Applications (TOSCA) defines several functional blocks, i.e., actors, playing different roles in the various phases of NS and VNF lifecycle management, from on-boarding to operations and monitoring.

[0004] FIG. 1 shows functional blocks in the NFV reference architecture proposed by ETSI. The current ETSI standard specifies the NFV reference framework, its functional blocks, their roles, their interfaces, and some NS and VNF-related operational flows.

[0005] However, the current ETSI standard does not provide network service design and deployment processes.

SUMMARY

[0006] There is provided a method for providing a Virtualized Network Function (VNF) according to Network Service (NS) requirements. The method comprises selecting an on-boarded VNF descriptor (VNFD) from a VNF catalogue, configuring parameters of the selected on-boarded VNFD according to the requirements of the NS and instantiating a VNF according to the configured on-boarded VNFD.

[0007] There is also provided a method for providing a Network Service (NS). The method comprises selecting an on-boarded NS Descriptor (NSD) from a NS catalogue, modifying NSD information of the selected on-boarded NSD and instantiating the NS according to the modified on-boarded NSD.

[0008] There is provided a system operative to provide a Virtualized Network Function (VNF) according to Network Service (NS) requirements, the system running in a cloud computing environment comprising a NFV Orchestrator (NFVO) and a VNF Manager (VNFM). The system is operative to select an on-boarded VNF descriptor (VNFD) from a VNF catalogue, configure parameters of the selected on-boarded VNFD according to the requirements of the NS and instantiating a VNF according to the configured on-boarded VNFD.

[0009] There is also provided a system operative to provide a Network Service (NS), the system running in a cloud computing environment comprising a NFV Orchestrator (NFVO). The system is operative to select an on-boarded NS Descriptor (NSD) from a NS catalogue, modify NSD information of the selected on-boarded NSD and instantiate the NS according to the modified on-boarded NSD.

[0010] There is provided a non-transitory computer readable media having stored thereon instructions for executing any one of the methods described herein.

[0011] The methods and systems provided herein present improvements over the ETSI standard.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] FIG. 1 is a diagram illustrating the NFV architecture proposed by ETSI.

[0013] FIG. 2 is a diagram illustrating NFV artifacts.

[0014] FIG. 3 is a flowchart of an NS management process according to an embodiment.

[0015] FIG. 4 is a flow diagram illustrating successive tailoring of VNFs and NSs.

[0016] FIG. 5 is a flowchart of a method according to an embodiment.

[0017] FIG. 6 is a flowchart of a method according to another embodiment.

[0018] FIG. 7 is a flowchart of a method according to another embodiment.

[0019] FIG. 8 is a flowchart of a method according to another embodiment.

[0020] FIG. 9 is a flowchart of a method according to another embodiment.

DETAILED DESCRIPTION

[0021] Various features and embodiments will now be described with reference to the figures to fully convey the scope of the disclosure to those skilled in the art.

[0022] Many aspects will be described in terms of sequences of actions or functions. It should be recognized that in some embodiments, some functions or actions could be performed by specialized circuits, by program instructions being executed by one or more processors, or by a combination of both.

[0023] Further, some embodiments can be partially or completely embodied in the form of computer readable carrier or carrier wave containing an appropriate set of computer instructions that would cause a processor to carry out the techniques described herein.

[0024] In some alternate embodiments, the functions/actions may occur out of the order noted in the sequence of actions. Furthermore, in some illustrations, some blocks, functions or actions may be optional and may or may not be executed. Optional blocks are generally identified via dotted lines.

[0025] Referring again to FIG. 1 which shows the NFV reference architecture 100 as proposed in the ETSI standard, the main functional module in the architecture is the NFV Management and Orchestration (NFV-MANO) 101, which is in charge of deployment, management, and orchestration of NSs. The NFV-MANO 101 consists of three functional blocks: the NFV Orchestrator (NFVO) 102, the VNF Manager (VNFM) 103, and the Virtualized Infrastructure Manager (VIM) 105. These functional blocks are referred to as actors.

[0026] The main responsibilities of the NFVO include the NS orchestration and lifecycle management, e.g., (1) on-boarding VNFs and NSs; (2) instantiation of NSs and
VNFM; and (3) NS policy management related to affinity/anti-affinity, scaling, fault, and performance.

Virtualized Infrastructure Manager (VIM) 105 manages and orchestrates the NFVI resources. The additional functionalities of the VIM include the allocation, upgrade, release, reclamation, usage optimization, capacity discovery, and capacity/usage reporting of physical and virtual resources. The VIM supports the management of VNF Forwarding Graphs (VNFFGs), which includes creating and maintaining virtual links (VLs). VIM also performs the NFVI physical and software resource repository management.

The NFV reference architecture includes other functional blocks, like the VNF 106, the EM 108, the NFVI 107, and the OSS/BSS 110 which are now described.

Virtualized Network Functions (VNFs) 106 are the building blocks of an NS in NFV. VNFs are software pieces with the same functionality as their corresponding physical network functions (PNFs), e.g., a virtual firewall (vFW) vs. a traditional firewall device. A VNF 106 can be composed of multiple internal components (VNFC). The description of the deployment and operational behavior of a VNFC is defined as a Virtual Deployment Unit (VDU). For each VNF, a manager (VNFM) 103 is assigned in the NFV framework to manage its lifecycle.

Element Management (EM) 108 carries out the fault management, configuration, accounting, performance, and security (FCAPS) functionality for one or multiple VNFs 106. The EM 108 configures a VNF to provide a specific network function and performs the fault management related to that network function; (2) manages accounting, e.g., billing, for the usage of VNF functions; (3) performs the security management for the VNF functions; (4) works together with the VNFM by exchanging information on NFVI resources to manage the virtualized resources for a VNF.

Network Function Virtualization Infrastructure (NFVI) 107 is the platform on which the network services are deployed. The NFVI has the physical layer of devices for computation, storage, and network; the virtualization layer, e.g., hypervisor; and the virtualized resources, e.g., virtual machines. The NFVI 107 is the totality of physical and virtual resources.

Operations Support Systems and Business Support Systems (OSS/BSS) 110 refer to the operator’s proprietary systems and management applications supporting their business. The OSS/BSS systems 110 exchange a lot of information with NFV-MANO 101 functional blocks to provide the desired network service. Moreover, in the NFV architecture, the OSS/BSS 110 provides management and orchestration of legacy systems.

During the lifetime of an NS, various artifacts 201 at various levels of abstractions are used and produced. The artifacts 201 can be divided into three categories: descriptors 202, records 207, and data repositories 212, as shown in FIG. 2.

Descriptors 202 describe the deployment requirements, operational behavior, and policies required by the NSs or VNFs. The virtual links (VLs) that connect VNFs to form a network topology also come with their descriptors (Virtual Link Descriptors (VLDs) 206). Similarly, the VNF Forwarding Graph (VNFFG) has a descriptor, i.e., VNFFGD 204, which defines an end-to-end sequence of VNFs that packets traverse. All these descriptors, e.g., NSDs 203, VNFs 205, VNFFGs 204, and VLDs 206, are known as deployment templates.

Records 207 describe individual instances of NSs, VNFs, VNFFGs, and VLs, in NSRs 208, VNFFGRs 209, VNFRs/PNFs 210 and VLRs 211 respectively. These records are stored within the NFV Instances Repository.

Data Repositories 212 that are defined in the NFV architecture are (1) the NS Catalogue 213 containing all the on-boarded NSDs, VNFFGDs, and VLDs; (2) the VNF Catalogue 214 containing all the on-boarded VNFs; (3) the NFV Instances Repository 215 containing all the instance records; and (4) the NFVI Resources Repository 216 holding information about the available, reserved, and allocated NFVI resources as updated by the VIM.

A workflow for NS and VNF deployment and management will be described and comprises a chaining of the different functions and operations defined in the standard. These functions and operations are grouped into activities. The relations and dependencies between these activities as well as the dependencies and the relations between the NFV reference framework functional blocks are explicitly defined in the workflow. The proposed workflow is compliant with the NFV reference framework as it obeys to the standard definitions of the functional blocks, roles, and interfaces.

Moreover, this process goes beyond the reference framework and proposes the automatic generation of a new NS Descriptor (NSD) and other artifacts from the tenant’s NS requirements and the catalogue of VNF Descriptors (VNFs) upon tenant request.

This workflow is a first step toward the necessary automation of network service design and deployment process for NFV systems. To achieve this, the following steps are followed: (1) investigate and identify the different functions and operations in relation to the functional blocks; (2) identify and capture various interactions among the functional blocks; (3) identify the limitations and open issues in the NFV reference framework; (4) determine how the different artifacts are refined throughout the process; (5) fill the gaps and define a concrete process, with high level activities for grouping together operations and functions defined in the standard, that is compliant to the NFV reference framework.

The ETSI standard includes a discussion on the initiation and deployment process of NSs and VNFs. As mentioned in the standard, a network service is requested by a tenant’s OSS/BSS 110. The NEVO 102 then on-boards the VNFs into the NFV Catalogue and the NSD, the VNFFGDs, and the VLDs into the NS Catalogue. During on-boarding, the NFVO checks the integrity of NSD, VNFFGD, VLD, and VNF/PNF. Once the descriptors have been on-boarded the NS can be instantiated on request.

In the course of the instantiation process, the NEVO 102 receives additional instantiation parameters from the entity (e.g., OSS/BSS 110) initiating the instantiation process. These instantiations input parameters contain information on deployment flavors and are used to customize a
specific NS instance and its VNF instances. The deployment flavors specify parameters for central processing unit (CPU) utilization and other factors related to the virtual machine on which the VNFs are deployed. The instantiation process results in various records, namely the Network Service Records (NSRs), the VNF Records (VNRs), the Virtual Link Records (VLRs), and the VNF/FG Records (VNF-FGRs). This is followed by the deployment of the NS and the associated VNFs.

[0043] Some limitations and open issues that are not addressed in the current NFV standard have been identified:

[0044] Although the NFV framework describes three functional blocks under MANO, they are typically not implemented separately due to lack of clear segregation of responsibilities during the instantiation and deployment phases.

[0045] The standard identifies various orchestration functions and operations along with operational flows that are part of the NS and VNF lifecycle management. The standard does not propose any logical grouping of these functions and operations.

[0046] The standard does not specify explicitly the dependencies among the various functional blocks in the NFV framework, i.e., how the functions and operations are interdependent to achieve the NFV goals.

[0047] As a consequence, there is no concrete process (or workflow) that defines the execution order of the various functions and operations. A detailed and concrete process model is required to put the reference framework into practice. It will enable the automation of network service deployment and management.

[0048] A vendor providing a VNF implementation may not want to restrict it in the associated VNF descriptor (VNFD) to a particular deployment option, but may want to allow for the widest possible range of options appropriate for the deployment. Compared to this, a given VNFD may limit the deployment options, for example, due to some policies or available resource of the operator/NFV provider. Furthermore, an NS may impose further limitations depending on the tenant’s policies and/or targeted key performance indicator (KPI) values. Considering that the same VNFM package may be used to implement different NSs with diverse policies and KPIs the use of the same VNFD can be problematic. Currently, the specifications do not distinguish these different stages and define only a single VNFD associated with each VNF to reflect these scenarios. A detailed process can distinguish these different stages and the different tailorings.

[0049] An NS design and deployment which is compliant to the NFV reference framework is therefore proposed. In this process, high level activities for grouping the different functions and operations, their dependencies and relationships are identified. The inputs, outputs of each activity and the tailoring of the artifacts are clearly specified. In addition, going beyond the standard, an activity is included for the automated design of the NSD from the tenant’s NS requirements (NSReq). This activity is referred to as the NS Design activity. This leads to an additional limitation of the current specifications:

[0050] The NFV specifications focus on the virtualization and software management aspects of the VNFs, e.g., the interfaces and KPIs. However, the current specifications do not define any method that would help in identifying the functionalities a VNF can provide for its users.

[0051] Turning to FIG. 3, the network service design, configuration and deployment process 300 will be described. The functions of the OSS/BSS 110 in the context of the proposed process are elaborated, and a workflow which is a chaining of the various NFV functions and operations grouped into activities required to carry out NS deployment is proposed. This work is based on the VNF as a Service (VNFaaS) model as defined in the ETSI standard. Further to that, the OSS/BSS 110, the EM 108, and other external subsystems (NFV1 resources repository, NFV instance repository, NS and VNF catalogues) are brought into the picture along with the NFV-MANO 101 to propose the new NS design and deployment process.

[0052] The activities that are part of the workflow are identified and described (based on the orchestration functions and operations defined in the ETSI standard) along with the dependencies (in terms of information elements) between the activities which sets a well-defined scope for each of them. The actor responsible for each of these activities, including the activity of NS design, is also identified. The NS design is a complex activity that is outside the scope of the NFV standard. Before presenting the whole process, NS design activity is described briefly.

[0053] In the proposed process, the policy management (e.g., scaling and threshold) and monitoring aspects of NS that are required to carry out service level agreement (SLA) monitoring, management and dynamic reconfiguration are abstracted and the focus put on the design and deployment of network services.

[0054] Network service design entails the definition of deployment templates (namely NSD, VNFFGD, and VLD), which includes static information elements related to an NS. This NSD is used by the orchestrator during the deployment.

[0055] There is proposed a method for NSD generation in which a tenant may request a new NS by specifying the NSReq, which consists of functional and non-functional requirements possibly with some initial decomposition targeting specific VNFs. It is assumed that the OSS/BSS of an NFV provider gets the NSReq and generates the NSD based on the provider’s VNF Catalogue. The NSD generation method involves the decomposition of the NSReq and the selection of proper network functions, e.g., VNFs and PNFs from the VNF Catalogue. The NSD management may also involve PNFD management such as PNFD on-boarding, update, deletion and query. A person skilled in the art would understand that the management of PNFDs is similar to that of VNFDs and is not described here.

[0056] The NSReq decomposition is guided by a network function ontology (NFO), which captures standard network function (de)compositions as defined by different standardization bodies such as the 3rd generation partnership project (3GPP) as well as knowledge and experience from previous decompositions. The NFO captures the decomposition of network functionalities to a level of granularity where each functionality can be mapped onto some VNF provided functionality. When the decomposition reaches that level, VNFs from the VNF Catalogue are matched and selected to compose the network service. The selection takes into account the non-functional requirements and the VNFD of the selected VNFs are tailored accordingly to NS specific VNFDs, referred to as NS-VNFDs. During this activity, the
VNF forwarding graph descriptor (VNFFGD) and the virtual link descriptors (VLD) are also generated.

Still referring to FIG. 3, the NS design and deployment process is described as a model-based workflow covering the phases from NS design to deployment. With the process, the gaps are bridged between the various functional blocks and the various activities that are part of NS design and deployment. Having a model-based process also allows to take advantage of the ETSI/TOSCA defined NFV profiles.

The Process Model (PM) illustrated in FIG. 3 shows the control-flow and data-flow among the actors involved in the process. In an embodiment, the PM language is a variant of the Formalism Transform Graph (FTG)+PM language and the PM is modelled using the unified modelling language (UML) 2.0 activity diagram formalism which is typically used to model software and hardware processes. The formalism allows the modelling of concurrent processes and their synchronization with the use of fork and join nodes. Both control-flow and object-flow can be depicted in the model. An activity node can either be a simple action (representing a single step within an activity) or an activity (representing a decomposable activity which embeds actions or other activities). An activity specifies a behavior that may be reused, i.e. an activity can be included in other activity diagrams to invoke behavior. For representation purpose and to add clarity to the entire process, activity diagram partitions (a.k.a., swim lanes) are used, where each partition represents one actor (participating functional block) in the process. Multicasting and multi-receiving are used in conjunction with partitions to model object flows between activities that are the responsibility of objects determined by a publish and subscribe facility. Object flows are tagged with “multicast” or “multi-receive” in this case. Since UML 2.0 activity diagrams are given semantics in terms of Petri Nets, the precise formal semantics allow the activity diagrams to be simulated and analyzed.

In the present PM, the artifacts are considered as models (instances which conform to existing meta-models) which are either inputs or outputs of activities.

The PM covering the design, configuration generation, and deployment activities is presented in details in FIG. 3. Along with the high-level activities, the PM clearly shows the input and output models associated with each activity via input and output pins (denoted by the tiny squares on the activity border). In the PM, thin edges denote object-flow, while thick edges denote control-flow. The join and fork constructs are represented as horizontal bars.

The role of each NFV functional block is shown as partitions in the PM. The partitions tagged as “external” denotes blocks and actors outside the NFV-MANO. The partition labelled as Other Blocks includes blocks in the NFV architecture that are considered as black-boxes in our work and which communicate with the other actors via predefined interfaces.

The activities, the associations between the different actors, and the dependencies between the activities, are discussed in more details below.

The activities (1) and (3) in the following list can be carried out concurrently (shown with the parallel flow coming out of the fork construct in FIG. 3). Also, activities in (1) and (2) are continuous activities and not part of the process for the NS design and deployment, which starts with the NS design activity triggered by a tenant’s 301 request for a new service. The steps of the process are now described.

(1) Update VNF Catalogue (actor: OSS/BSS). The vendor 302, an external actor, sends a new VNF package (including the VNFD and the VNF Implementation). The VNF package is received by the OSS/BSS 110 (also an external actor) and the OSS/BSS VNF Catalogue maintained by the OSS/BSS is updated. This is a continuous behavior which accepts a stream of incoming VNF packages from vendors and updates the internal catalogue. To show streaming, the text annotation stream is placed near the (input/output) pin symbol in the PM.

(2) On-boarding VNF (actor: NFVO). The NFVO 102 takes the VNF specification and configuration data that is part of the incoming VNF package from the OSS/BSS VNF Catalogue to on-board the VNFs. The NFVO 102 validates the VNFD and updates the VNFCatalogue (maintained by NFV-MANO) with the On-boarded-VNFD, which is the vendor provided VNFD tailored by NFVO according to the constraints and policies of the given NFV system. After that, the NFVO 102 requests VIM 104 to upload the VNF image (triggering the Upload Image action). The on-boarding of a VNF is required prior to its deployment. On-boarding is a continuous behavior which periodically receives vendor-provided VNF packages from the OSS/BSS and streams out the On-boarded-VNFD model for each incoming VNF. At this point, the On-boarded-VNFs are still generic with respect to the NSs, as in they are not associated with any NS. The VNFs are made NS specific in the NS Design activity.

(3) NS Design (actor: OSS/BSS). The tenant 301, an external actor, requests a new network service by providing a new NSReq which triggers the whole process starting with the NS design activity within the OSS/BSS 110. The activity also takes as input the NFV-MANO VNF Catalogue consisting of On-boarded-VNFs, to produce an NSD according to the available VNFs. The deployment templates including the NSD, VNFFGD, and VLD to be deployed are created during this activity as described in Section 4.1. The On-boarded-VNFs (from (2)) are specialized for this NS resulting in NS-VNFs. An additional model, NS-NFR is also output and contains the non-functional requirements (NFR) specified in the NSReq that need to be carried forward and addressed during the configuration generation activity.

(4) On-boarding NS (actor: NFVO). Following the NS design, the NFVO 102 continues with on-boarding the NS. On-boarding entails validating the NSD, VNFFGD, and VLD to check for missing elements and updating the NS Catalogue. This activity tailors the deployment templates resulting in the artifacts On-boarded-NSD, On-boarded-VNFFGD and On-boarded-VLD. The NS-specific VNFD(s) and the NFRs are received from the OSS/BSS 110 and forwarded to the corresponding VNFM 103 through the NEVO 102.

(5) VNF Configuration Generation (actor: VNFM/EM). Following the NS on-boarding by the NFVO 102, the NS-specific VNFs need to be configured with deployment-specific details (based on their deployment templates) to proceed with deployment. The internal configuration of each VNF is generated as part of this activity. The internal connections between the VNF components (VNFc) are also configured. A configuration generation request (along with the NS-VNFD and the NS-NFR) is received from the NFVO
102 by each VNFM 103 which may involve the EM 108 in the activity. Each configuration generation includes the generation of the VNF InternalConfig and the VNF record (VNFR). The internal configuration may determine what the VNF configuration should be and how the VNFRs should be created. The NS-NFR is used to determine the number of instances of VNFs. The records created are stored in the NFV Instance Repository. The VNFM 103 respond by returning the VNFRs created to the NFVO 102, which is done via multi-receive, i.e., the objects in the flow are gathered from respondents to multica sting.

[0069] (6) NS Configuration Generation (actor: NFVO). The deployment-specific details of an NS are now generated to enable NS deployment. Once the VNFRs are created for the VNFs, the NS configuration activity is triggered by the NFVO 102 (once all VNFR input models for the associated VNFs are available for the activity). The external links (VLs) between the VNFs are configured during this stage. The instantiation takes into account the constraints defined in NS-NFR. The records created (e.g., VNFFGR, VLR and NSR) representing the NS instances along with the refined VNFRs are saved in the NFV Instance Repository.

[0070] (7) Initiate NS Deployment (actor: NFVO). The NFVO 102 then initiates the NS deployment. This begins with the deployment of the VNFs that are part of the NS. The NFVO 102 checks for resource availability for each VNF. The resource reservation for the VNFs can also optionally be carried out by the NFVO 102 via the VIM 105. Once the resources are reserved, the NFVO 102 sends the reservation acknowledgment to the corresponding VNFM 103 (by multicasting the VNFM 103). The VIM 105 identifier is also passed to inform the VNFM 103 where to deploy the VNF. This triggers the deployment of the VNFs by the VNFM 103. It is also possible that OSS/BSS 110 controls the deployment and this activity is only triggered upon its request.

[0071] (8) VNF Deployment (actor: VNFM/EM). Once the resource reservation is completed in activity (7), each VNFM 103 communicates with the VIM 105 for resource allocation. The VNF Resource Provisioning and Interconnection Setup action is triggered in the VIM 105. The VNFM 103 then configures the VNF with deployment-specific parameters (VNF-specific lifecycle parameters), which involves instantiation and configuration of the VNF components according to the VNFR. The EM 108 is involved with the configuration of VNF with application-specific parameters to complete the actual VNF deployment. At this point the VNFR is updated as well as the NFVI Instance Repository. Once all the VNFs are deployed, the NFVO 102 continues with the NS deployment.

[0072] (9) Complete NS Deployment (actor: NFVO). Following the deployment of the VNFs, the NFVO 102 continues to deploy the NSR, VLR, and VNFFGR by requesting the VIM 105 for network connectivity creation. The NS Resource Provisioning and Interconnection Setup action is triggered at this point. Upon confirmation from the VIM 105 of successful creation of network connectivity, the NFVO 102 requests the VIM to connect the VNFs to the network (setting up the VLs that are part of the VNFFG(s)). The deployment parameters in the records are updated as well as the NFVI Instance Repository.

[0073] According to the ETSI standard, it is possible to initiate resource allocations in different ways. For this reason, in the standard there are several variants of the operational flow for VNF and NS deployment (depending on whether the resource allocation is initiated by the NFVO 102 or the VNFM 103). Therefore, it should be noted that it is possible to have variations in the workflow that is depicted in the PM depending on the interaction model chosen.

[0074] Following deployment, the NS instance management continues until it is decommissioning.

[0075] In the NFV domain, various artifacts are used. However, with respect to the VNFD, there could be a substantial shift from what is initially provided by the VNF vendor and the specific version that is deployed at the end for a specific tenant.

[0076] The current NFV specification does not discuss the different specializations an artifact goes through during the NS design and deployment process. The NFV artifacts, for example the VNFD and NSD, get transformed through a chain of activities as shown in FIG. 4, which illustrates different tailoring of the VNFD and the NSD to complement the PM. Artifacts (other than VNFD and NSD), such as the VLDs, the VNFFGds, and the various catalogues/repositories can also be tailored.

[0077] VNFD Tailoring 400 is now discussed in relation to the top part of FIG. 4. An initial Vendor-provided VNFD 401 is generic, i.e., open to many deployment options of the VNF, and includes the VNF identification data (e.g., ID, vendor, and version), VNF-specific data (e.g., connectivity requirements, inter-dependencies of VNFs, and deployment flavors), VNFC data (e.g., specific VNFC configuration data and deployment constraints), and virtualized resource requirements (step 1). For the On-boarded-VNFD 402, the orchestrator checks for any missing mandatory information elements in the VNFD and provides with the default values, if required according to the policies and constraints of the given NFV system, which is the first tailoring of a VNFD (step 2). The NS-specific VNFDs 403 are the result of VNF selection from the VNFCatalogue and their tailoring for a specific NS based on the NSReq from the tenant where various VNF/VNFC deployment constraints and virtualized network resource requirements might be added/updated in the VNFD by the OSS/BSS (step 3). Once the VNFDs are tailored to be NS-specific, the VNFM/EM configures VNF-specific data for VNF instantiation. The VNFR 404 is created and the VDU-level configuration is performed. Additional information elements are introduced in the VNFR, for instance, information on the parent NS, the logging capabilities for audit purpose, the network address, and the VNFM managing this instance (step 4). A further tailoring of the VNFR is possible based on the deployment flavor and geographical location constraint, which results in NS-specific VNFR 405 (step 5). During the deployment of the VNFs, the policies, e.g., affinity and anti-affinity rules, are applied between VMs and hosts, which results in the refinement of VNFRs (deployed VNFR 406) to adhere to the policies (step 6).

[0078] NSD Tailoring 400 is now discussed in relation to the bottom part of FIG. 4. The Generated NSD 407 includes the (On-boarded) NS-specific VNFDs 403, the VNFFGds, and the VLDs (step 1). For the On-boarded NSD 408, the NFVO checks for any missing mandatory information elements in the NSD and provides with the default values, if required according to the policies and constraints of the given NFV system, which is the first tailoring of an NSD (step 2). Later, the NFVO instantiates the NS by creating a network service record (NSR) 409 by adding additional information to the On-boarded-NSD 408, for instance, infor-
mation on NFVI resources reserved for this NS instance, the NS status, and the logging capabilities for audit purpose (step 3). During the deployment of an NS, various policies (e.g. the NS scaling policy) are provided from OSS/BSS resulting in a further refinement of an NSR (deployed NSR 410) (step 4).

[0079] The PM described above may have workflow variations, i.e. the NSD can be designed from scratch or an existing NSD can be reused based on the new functional/non-functional requirements from tenants. The proposed PM can be enacted using the benefits of model-driven engineering with a final goal to automate the workflow for designing and deploying an NS.

[0080] In one embodiment, the proposed process covers the design, configuration and deployment of a new network service triggered by a request from a tenant in the form of an NSReq consisting of functional and non-functional requirements. The NS Design activity can include a function for checking the NS Catalogue for a service that meets the NSReq before starting the design and reusing what is already available. However, if the non-functional requirements are taken into account in the design of the NS, the probability of a matching NSD is low. One can think of an alternative workflow where the NSReq consists only of functional requirements to improve NS re-usability. In this case the NS is designed with multiple versions in a first step and the non-functional requirements are taken into account only in a second step where a specific version that can meet the non-functional requirements is selected and tailored further to meet the non-functional requirements. Different tenants with different non-functional requirements may reuse the same NSDs. The provider can reuse existing network services (NSDs) to configure and instantiate network services based on varying non-functional requirements.

[0081] In another embodiment, the process is triggered by a new network service request from a tenant. From then on, the process can be executed automatically. It is also possible that, instead of the different phases of the process can be initiated by the OSS/BSS. Following NS on-boarding, the process can be suspended and resumed only when the OSS/BSS sends an instantiation and deployment request.

[0082] In another embodiment, using model-driven engineering (MDE) and model-based engineering (MBE), software models can play a role in the lifecycle management of a system. Having an effective and efficient way of coordinating these models, maintaining consistency, and managing the propagation of changes across the models. As a step in that direction, domain and processes are modeled using modeling languages that are well-established and have well-defined semantics, such as UML 2.0 Activity Diagrams. The use of such modelling language sets the pillars for model-driven process enactment or execution.

[0083] FIG. 5 presents a flowchart of a method 500 executed by a NFV Management and Orchestration (NFV-MANO), the NFV-MANO being executed in a cloud computing environment which provides processing and interface circuits and memory for running the NFV-MANO, the memory containing instructions executable by the processing circuits whereby the NFV-MANO is operative to:

[0084] receive, step 501, a request for a new network service (NS);

[0085] on-board, step 502, the NS;

[0086] initiate, step 503, virtualized network functions (VNFs) configuration generation by sending a configuration generation request to at least one VNF manager (VNFM);

[0087] receive, step 504, VNF records (VNFRs) created by the VNFM;

[0088] generate, step 505, the NS configuration;

[0089] initiate, step 506, deployment of the NS; and

[0090] complete, step 511, deployment of the NS.

[0091] According to the method, the request for a new network service is received by an operations support system and business support system (OSS/BSS) and includes a new network service requirements (NSReq), which triggers NS design activity within the OSS/BSS and wherein a network function virtualization management and orchestration (NFV-MANO) VNF Catalogue, consisting of on-boarded VNFDs 402, is used to produce an NS descriptor (NSD) according to available VNFs.

[0092] According to the method, deployment templates including the NSD, VNF Forwarding Graph (VNFFG) descriptor (VNFFGD) and Virtual Link Descriptor (VLD) to be deployed are created and wherein an NS-NFR containing the non-functional requirements (NFR) specified in the NSReq is outputted.

[0093] According to the method, on-boarding comprises validating the NSD, VNFFGD, and VLD to check for missing elements and updating the NS Catalogue.

[0094] According to the method, on-boarding tailors/generates a deployment templates resulting in an On-boarded-NSD artifact, an On-boarded-VNFFGD artifact and an On-boarded-VLD artifact.

[0095] According to the method, NS-specific VNF descriptor(s) (VNFD(s)) and the NFRs are received from the OSS/BSS and forwarded to corresponding VNFM(s).

[0096] According to the method, a VNFM configures NS-specific VNFs with deployment-specific details.

[0097] According to the method, an internal configuration of each VNF is generated and internal connections between the VNF components (VNFC) are configured by the VNFM and element management (EM).

[0098] According to the method, the configuration generation includes the generation of a VNFInternalConfig and a VNFR and wherein records created are stored in the VNFR Instance Repository.

[0099] According to the method, the VNFM respond by returning the VNFRs created to the NFVO, which is done via multi-receive, i.e. the objects in the flow are gathered from correspondents to multicasting.

[0100] According to the method, external links (VLs) between the VNFs are configured and instantiation takes into account the constraints defined in NS-NFR.

[0101] According to the method, deployment of the VNFs that are part of the NS comprises:

[0102] checking, step 507, for resource availability for each VNF;

[0103] initiating, step 508, reserving of resource for the VNFs;

[0104] sending, step 509, reservation acknowledgment to corresponding VNFM(s); and

[0105] sending, step 510, VIM identifier to inform VNFM of where to deploy each VNF, thereby triggering deployment of the VNFs by the VNFM(s).

[0106] According to the method, the above four steps are executed by the NFVO.
According to the method, once the resource reservation is completed:

- the VNFM communicates with the VIM for resource allocation;
- the VNFM triggers the VNFR to Resource Provisioning and Interconnection Setup action in the VIM;
- the VNFM configures the VNFR with deployment-specific parameters (VNFC-specific lifecycle parameters), which involves instantiation and configuration of the VNFR components according to the VNFR;
- the element management (EM) configures the VNFR with application-specific parameters to complete the actual VNFR deployment.

According to the method, the NFVO is further operative to:

- send, step 512, a request to the virtualized infrastructure manager (VIM) for network connectivity creation;
- receive, step 513, a confirmation from the VIM of successful creation of network connectivity; and
- request, step 514, the VIM to connect the VNFR to a network.

FIG. 6 presents a flowchart of an embodiment 600 as described previously, wherein VNFR Tailoring is executed during the on-boarding step, starting with a generic Vendor-provided VNFR including VNFR identification data, such as identifier (ID), vendor, and version, VNFR-specific data, such as connectivity requirements, inter-dependencies of VNFRs, and deployment flavors, VNFR data, such as specific VNFR configuration data and deployment constraints, and virtualized resource requirements, the VNFR tailoring comprising:

- providing, step 601, default values for any missing information elements in the VNFR as well as applying policies and constraints of the given NFV system;
- selecting, step 602, VNFRs from the VNFR Catalogue;
- tailoring of VNFRs, step 603, corresponding to the selected VNFRs, for a specific NS based on the NSReq from the tenant where various VNFR/VNFC deployment constraints and virtualized network resource requirements might be added/updated in the VNFR by the OSS/BSS;
- the VNFR/EM configuring, step 604, VNFR-specific data for VNFR instantiation;
- creating, step 605, the VNFR; and
- performing, step 606, the virtual deployment unit (VDU)-level configuration.

According to the method, additional information elements are introduced in the VNFR, such as, information on the parent NS, logging capabilities for audit purpose, network address, and an identification of the VNFR managing this instance.

According to the method, during deployment of the VNFRs, policies, such as affinity and anti-affinity rules, are applied between VMs and hosts.

FIG. 7 presents a flowchart of embodiment 700 as described previously, wherein NS Tailoring is executed, and wherein a Generated NSD includes the on-boarded NS-specific VNFRs, the VNFFGIDs, and the VLDs, the NS tailoring comprising:

- the VNFR communicates with the VIM for resource allocation;
- the VNFR triggers VNFR Resource Provisioning and Interconnection Setup action in the VIM;
- the VNFR configures the VNFR with deployment-specific parameters (VNFR-specific lifecycle parameters), which involves instantiation and configuration of the VNFR components according to the VNFR;
- and the element management (EM) configures the VNFR with application-specific parameters to complete the actual VNFR deployment.

FIG. 8 illustrates a method 800 for providing a Virtualized Network Function (VNF) according to Network Service (NS) requirements. The method comprises selecting, step 801, an on-boarded VNF descriptor (VNFD) from a VNF catalogue, configuring, step 802, parameters of the selected on-boarded VNFD according to the requirements of the NS, and instantiating, step 803, a VNF according to the configured on-boarded VNFD.

The method may be executed in a Network Functions Virtualization (NFV) system, running in a cloud computing environment, comprising a NFV Orchestrator (NFVO) on-boarding the VNFR in the VNF catalogue and configuring the parameters of the selected VNFR, and a VNF Manager (VNF/M) instantiating the VNFR.

The method may alternatively be executed in a Network Functions Virtualization (NFV) system, running in a cloud computing environment, comprising a NFV Orchestrator (NFVO) on-boarding the VNFR in the VNF catalogue, an Operations Support Systems and Business Support Systems (OSS/BSS) configuring the parameters of the selected VNFR, and a VNF Manager (VNF/M) instantiating the VNFR.

The VNFR may comprise VNF identification data, VNFR-specific data, VNFC data, and virtualized resource requirements. The VNF identification data may comprise an identifier (ID), a vendor, and a version. The VNFR-specific data may comprise connectivity requirements, inter-dependencies of VNF components (VNFCs), and deployment flavors. The VNFR data may comprise data such as specific VNFC configuration data and deployment constraints. Configuring parameters may comprise providing values for information elements in the VNFR and the values may be default values. Configuring parameters may further comprise updating the VNFR with deployment constraints and virtualized network resource requirements. In some embodiments, the VNFR may be duplicated or copied, before the updating step, to keep a copy of the original VNFR.

The method may further comprise creating, step 804, a VNF Record (VNR) and the NFVO may create the VNF Record (VNR). The VNF may update the VNR with further configuration information.

FIG. 9 illustrates a method 900 for providing a Network Service (NS). The method comprises selecting,
step 901, an on-boarded NS Descriptor (NSD) from an NS catalogue, modifying, step 902, NSD information of the selected on-boarded NSD and instantiating, step 903, the NS according to the modified on-boarded NSD.

[0137] The method may be executed in a Network Functions Virtualization (NFV) system, running in a cloud computing environment, comprising a NFV Orchestrator (NFVO) on-boarding the NSD in the NS catalogue and configuring the parameters of the selected NSD. The modifying information may comprise providing values for information elements in the NSD and the values may be default values. The information may comprise information related to constraints or policies of the NFV system.

[0138] The method may further comprise creating, step 904, an NS Record (NSR) and may also comprise adding, step 905, Network Function Virtualization Infrastructure (NFVI) resources reserved for an NS instance of the NS, an NS status, and a logging capability for audit purpose to the NSR.

[0139] Returning to FIG. 1, there is provided a system 100 operative to provide a Virtualized Network Function (VNF) 106 according to Network Service (NS) requirements, the system running in a cloud computing environment comprising a NFV Orchestrator (NFVO) 102 and a VNF Manager (VNFM) 103. The system 100 is operative to select an on-boarded VNF descriptor (VNFD) from a VNFC vault, configure parameters of the selected on-boarded VNFD according to the requirements of the NS and instantiate a VNF according to the configured on-boarded VNFD.

[0140] Still referring to FIG. 1, there is provided a system 100 operative to provide a Network Service (NS), the system running in a cloud computing environment comprising a NFV Orchestrator (NFVO) 102. The system is operative to select an on-boarded NS Descriptor (NSD) from an NS catalogue, modify NSD information of the selected on-boarded NSD and instantiate the NS according to the modified on-boarded NSD.

[0141] Modifications and other embodiments will come to mind to one skilled in the art having the benefit of the teachings presented in the foregoing description and the associated drawings. Therefore, it is to be understood that modifications and other embodiments, such as specific forms other than those of the embodiments described herein, are intended to be included within the scope of this disclosure. The described embodiments are merely illustrative and should not be considered restrictive in any way. The scope sought is given by the appended claims, rather than the preceding description, and all variations and equivalents that fall within the range of the claims are intended to be embraced therein. Although specific terms may be employed herein, they are used in a generic and descriptive sense only and not for purposes of limitation.

1. A method for providing a Virtualized Network Function (VNF) according to Network Service (NS) requirements, comprising:
   selecting an on-boarded VNF descriptor (VNFD) from a VNFC vault;
   configuring parameters of the selected on-boarded VNFD according to the requirements of the NS; and
   instantiating a VNF according to the configured on-boarded VNFD.

2. The method of claim 1, wherein the method is executed in a Network Functions Virtualization (NFV) system, running in a cloud computing environment, comprising a NFV Orchestrator (NFVO) on-boarding the VNFD in the VNFC vault and configuring the parameters of the selected VNFD, and a VNF Manager (VNFM) instantiating the VNF.

3. The method of claim 1, wherein the method is executed in a Network Functions Virtualization (NFV) system, running in a cloud computing environment, comprising a NFV Orchestrator (NFVO) on-boarding the VNFD in the VNFC vault, an Operations Support Systems (OSS/BSS) configuring the parameters of the selected VNFD, and a VNF Manager (VNFM) instantiating the VNF.

4. The method of claim 1, wherein the VNFD comprises VNF identification data, VNF-specific data, VNFC data, and virtualized resource requirements.

5. The method of claim 4, wherein VNF identification data comprises an identifier (ID), a vendor, and a version.

6. The method of claim 4, wherein VNF-specific data comprises connectivity requirements, inter-dependencies of VNF components (VNFCs), and deployment flavors.

7. The method of claim 4, wherein VNFC data comprises data such as specific VNFC configuration data and deployment constraints.

8. The method of claim 4, wherein configuring parameters comprises providing values for information elements in the VNFD.

9. The method of claim 4, wherein the values are default values.

10. The method of claim 9, wherein configuring parameters further comprises updating the VNFD with deployment constraints and virtualized network resource requirements.

11. The method of claim 1, further comprising creating a VNF Record (VNFR).

12. The method of claim 2, wherein the NFVO creates a VNF Record (VNFR).

13. The method of claim 11, wherein the VNFM updates the VNFR with further configuration information.

14. A method for providing a Network Service (NS) comprising:
   selecting an on-boarded NS Descriptor (NSD) from a NS catalogue;
   modifying NSD information of the selected on-boarded NSD; and
   instantiating the NS according to the modified on-boarded NSD.

15. The method of claim 14, wherein the method is executed in a Network Functions Virtualization (NFV) system, running in a cloud computing environment, comprising a NFV Orchestrator (NFVO) on-boarding the NSD in the NS catalogue and configuring the parameters of the selected NSD.

16. The method of claim 15, wherein modifying information comprises providing values for information elements in the NSD.

17. The method of claim 16, wherein the values are default values.

18. The method of claim 17, wherein the information comprises information related to constraints or policies of the NFV system.

19. The method of claim 14, further comprising creating a NS Record (NSR).

20. The method of claim 19, further comprising adding Network Function Virtualization Infrastructure (NFVI) resources reserved for an NS instance of the NS, an NS status, and a logging capability for audit purpose to the NSR.
21. A system operative to provide a Virtualized Network Function (VNF) according to Network Service (NS) requirements, the system running in a cloud computing environment comprising a NFV Orchestrator (NFVO) and a VNF Manager (VNFM), the system being operative to:
   select an on-boarded VNF descriptor (VNFD) from a VNF catalogue;
   configure parameters of the selected on-boarded VNFD according to the requirements of the NS; and
   instantiate a VNF according to the configured on-boarded VNFD.

22. A system operative to provide a Network Service (NS), the system running in a cloud computing environment comprising a NFV Orchestrator (NFVO), the system being operative to:
   select an on-boarded NS Descriptor (NSD) from a NS catalogue;
   modify NSD information of the selected on-boarded NSD; and
   instantiate the NS according to the modified on-boarded NSD.

23. (canceled)