Abstract

The emergence and continuous evolution of OTT applications have drastically changed the telecommunication ecosystem. These applications push for more versatile communications with dynamic behaviors and more constraints on the network. However, traditional Telco architectures are ill-suited to meet the dynamic requirements of today's application services. Network services are monolithic and vertically integrated which makes management operations complex and results in static networks unable to respond as applications requirements rapidly change. Telcos need to transform their network architectures for such variations in customer demands. The objective is to achieve flexibility in offering services as service customization is increasingly presented as the desired property to be integrated in future network services, and dynamicity in building, deploying and delivering network services on-demand. For that, we rely on paradigms and technologies on service and network virtualization and softwarization to propose transformation guidelines. We present service and network architecture principles and models along with deployment and management models as required elements to achieve a flexible and dynamic Network-as-a-Service architecture for Telcos.

Key words: X-as-a-Service, virtualization, Cloud Computing, Network Functions Virtualization, Software-Defined Networking, exposition, flexibility, dynamicity, Network-as-a-Service.

Contents

List of Figures

List of Tables

1 General Introduction

1.1 Context and Motivations

In recent years, some important trends have been accelerated causing the telecommunication ecosystem to change drastically. Some of the most important changes are at the service level.

The main trend reshaping the service layer ecosystem is the development of an entirely new type of Service Providers, the “Over-The-Top (OTT)" players. The rise of OTT players has driven an explosion of innovative applications that gain strong attraction by end users. These applications include Cloud and Content services for live and on-demand real-time video and communication applications. OTT applications became the new voice and messaging applications with hundreds of millions of users. In addition, the emergence of new smarter devices and faster access technologies (LTE, optic fiber ...) is leading to a rapid rise of these services. A vast spectrum of services such as M2M, smart city, smart home and e-health services, are being developed tending to make digital and connected services for almost all aspects of the society only by using IP connectivity networks.

However, as these new actors do not own network infrastructures., the data traffic generated from the explosion of OTT applications is faster increasing and pushes network operators to put huge investments to scale the network capacities and support the traffic volumes. Indeed, OTTs set up on top of operator networks without any interaction of any kind with this latter and provide these application services only by exploiting the universal reachability of the Internet.

With this multiplication of services and devices in an “any-to-any" communications style, OTT applications will continue to impose a huge throughput demand, pushing networks to their limits and operators to continuously invest to manage, scale and maintain their networks. Moreover, these services engender completely new ways of using application services usually totally unpredictable by the network that network operators have to deal with. They also push for more bandwidth consumption and for increasing demands for better Quality of Service (QoS) and Quality of Experience (QoE).

Therefore, operator networks are being used by OTTs as transport supports reducing operators' role to transport channels providers. OTTs provide data, voice and video application services similar in several cases to operators' legacy call and video services. So, Telcos ground is being increasingly preempted by OTTs. They are seeing their role shrinking into "dumb-pipe" providers carrying applications traffic from the so-called OTT actors.

Network operators need then to face a double-source competition; at the service level in front of OTTs and at both, at service and network levels, in front of network operators. The Telco value chain is no more exclusive for network service providers, operators and vendors but includes new Internet actors which go from application developers and providers to the biggest Cloud and Content services providers. The relationship between Telcos and OTTs is widely debated. Do OTT players represent a threat or an opportunity for network operators? Should we focus on the competition or move towards cooperation strategies? We believe both viewpoints are true. The threat is a motivating opportunity for Telcos to move towards new network and communication models.

All these facts lead to an ever-changing ecosystem where each actor is trying to exist. Ensuring a comfortable position within the new telecommunications ecosystem is one of the main motivations. Operators are seeking to maintain the role they traditionally have in the service delivery chain. For that, tending to preserve a direct linkage with the end clients to ensure an intermediation role between OTTs and end users using some key network assets seems to be a business guideline.

In this context, network operators' aim at monetizing their network by providing value-added network services, either using existing assets and thus improving investment profitability, or designing new ones in a manner that network services can be offered as tailored services in order to provide adaptable network solutions to these evolving service needs.

In front of the evolution of the application-layer, the interactions between network operators and OTTs to provide network services is no more suitable in today's environment and cannot allow network operators to maintain a strong position within the new value chain. Network operators need to establish new road maps in order to be aligned with these new business needs. This is a motivation for Telcos to set up new business and technological goals aligned with these new needs.

Beyond the business and strategy motivations, network operators are facing serious challenges to provide differentiated network services to meet the requirements of today's network users and address their various application needs. Network operatos need to rethink their network service delivery models and the technologies they use as we explain in the next Section.

1.2 Challenges and Objectives

As we have introduced, with the evolution to Cloud services, communications are undergoing a major transformation where the boundaries between IT, Cloud and network are disappearing to provide all capabilities and almost everything “as-a-Sevice".

The tendency is that application services and end users have dynamically changing needs in terms of network resources and services. These new services push users to have differentiated behaviors and event-related applications use. They expect ubiquitous access to services, from any device, via any access network. Mobility (user mobility, network mobility, and session mobility) are expected to be enabled by default, as devices become network agnostic capable of connecting to any available network.

Knowing that legacy network services are designed for and deployed on dedicated mobile-enabled infrastructures, this demands for ubiquitous services drives significant shift away from yesterday’s network architectures and service delivery models. Also, these application-level needs require new networking paradigms with new network control solutions which implies developing and deploying new network services and mechanisms.

However, all what the network operators can offer today is network connectivity and network transport networks within monolithic solutions, i.e. without any differentiation in network services, service delivery or the offered QoS. Operators offer the same network infrastructures and the same network services for all third-parties actors. Realizing the objective of providing third-party actors with customized network services is yet constrained by traditional network architectures.

Indeed, traditional operator network architectures are ill-suited to meet the dynamic requirements of today’s network users (enterprise, carriers, service providers, end users...). Current solutions and network architectures do not allow network operators to offer the flexibility to choose the most suitable network behavior needed for the applications and even less to do it dynamically on-demand and “on-the-fly".

In addition, the traditional way of implementing networks and services in hardware makes them static, rigid, manual and unable to respond as applications requirements rapidly change. Current network reactivity to changes in service needs is very slow because of vendor dependency, and network operations such as VPNs creation and modification or channel establishments (e.g. MPLS channel) are so complex and often requiring human intervention and fastidious (re)configuration through CLIs.

This static nature of the network makes it difficult for operators to offer adapted solutions. Based on their profiles and their core business, a service developer, an application service provider or enterprise have increasingly diverse requirements that monolithic network services in operator networks cannot address.

The objective of offering customized network services is coupled with the objective of building rapid responses to application needs. Operators have to be very agile in regard to network assets use and service development and deployment. They need to be extremely reactive to follow up with the fast services evolution that uses the Internet dynamic service model, especially that OTTs strategy is to have extremely short service deployment cycles and Time-To-Market using software-based approaches. Indeed, reducing network services development and deployment cycles is capital to boost operators' reactivity. They are working to rely on technologies that are most likely to help facing vendor lock-ins problems.

For operators to be aligned with the dynamicity of today's service applications and the unpredictable customer's demands, major network transformations are required. Operator network architectures need first to be leveraged from today's closed, complex and static networks where network services rely on monolithic platforms, with heavy configurations and protocol designs and strong equipment vendor dependency, to a flexible, dynamic and easy to manage network architectures. They need of course to meet the connectivity and QoS requirements but need also a huge flexibility and dynamicity to answer application services and third party actors' demands.

The main objectives is to provide flexible mechanisms for network service offering, to be able to dynamically deploy and provision network services, to ensure smooth service continuity, and to automate network services and network resources management. Furthermore, in order to ensure sustainable network transformations, operators should conceive their networks in a way that third-parties can have a self-service and open access to operator network capabilities.

First, the monolithic network services in existing network architectures need to consider some granularity aspects regarding their structure at the development phases. This way the new network services that need to be added can be developed then deployed using reusable network service components.

Then, relying on virtualized infrastructures as well as emerging virtualization paradigms and software-based network services can help to cope with vendor dependency issues and may contribute in reducing development and deployment cycles. At run-time, these technologies should facilitate the control and management of network services and resources.

Further, high automaticity of the network becomes essential especially for virtualized network services. Adopting the use of Application Programming Interfaces (APIs) and automation of network service deployment, control, provisioning and management would help to achieve the desired network dynamicity.

These network transformations that we consolidate under “Network Openness" would allow operators to address the technical needs and thus the business needs.

1.3 Thesis Scope

As detailed above, the needs for network openness are motivated by the need for flexibility and dynamicity.

Flexibility is necessary for the network service offerings. Indeed, even if operators might be able to anticipate the service requirements for a certain range of services, the specific demands of all the varieties of services cannot be accurately pre-determined when designing and deploying network services and infrastructures. Flexibility targets the ability to customize network services. Network operators should be able to have the flexibility to offer differentiated network solutions and propose different types of Service Level Agreements (SLAs).

The need for network openness is also a need for dynamicity. Dynamicity is necessary to allow network operators to efficiently react to satisfy the diverse application demands. A network provider should be able to respond dynamically to the changing needs with customized solutions. Thus, dynamicity targets the possibility to provide a network service on-demand.

The main thesis objective is to study the means for network openness where the network can be considered and offered as a service. Therefore, our objective is to achieve Network-as-a-Service. We have identified the technologies that are needed in order to reach network openness. Then, investigating how these objectives of flexibility in offering network services and dynamicity in delivering network services can be achieved and what are the impacts on the design, operation and management of Telco network services and network architectures.

The scope of the thesis and the main issue are delimited by technologies and paradigms from service and network architecture research areas. Based on the challenges presented above, they are most likely to be enablers for our openness objectives.

Figure 1.1 illustrates the thesis scope.

Today's monolithic network services should be conceived considering an adequate level of granularity to allow service reuse and mainly service customization through service composition. In this direction, component-based approaches and Service-Oriented Architecture (SOA) for Cloud services and applications development with the "as-a-Service" paradigm, constitute strong bases for our service customization objectives. So, the first area that we consider to defines the thesis scope is the Everything as-a-Service (X-as a-Service) area.

Virtualization has been an enabler for Cloud services and as-a-Service. Therefore, we consider virtualization in networks as a second area. We have identified Network virtualization as essential in dealing with network openness challenges. Network virtualization allows to build different logical network typologies on top of the same physical network infrastructure. We also talk about network slicing in more recent network architectures.

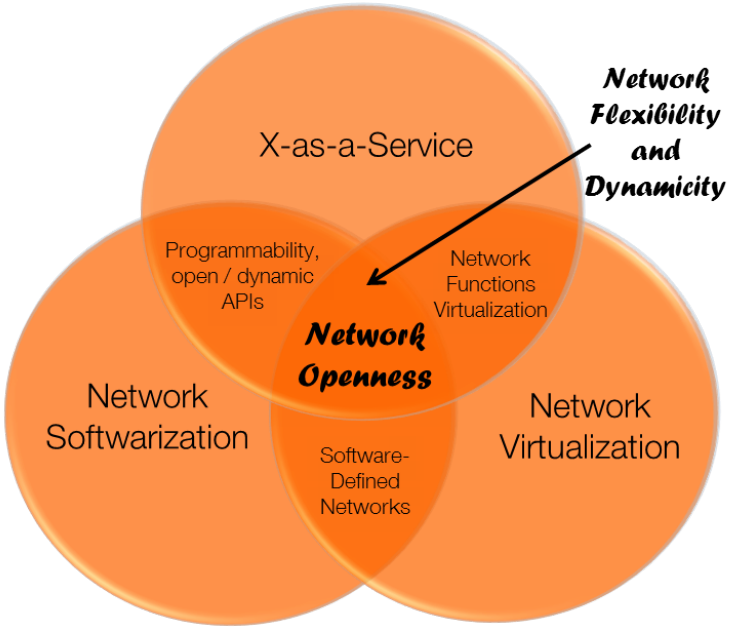

Figure 1.1 – Thesis Scope

We consider different virtualization technologies with their associated abstraction levels within a network architecture model. The choice of this concept is mainly motivated by the problems related to physical network infrastructure resource utilization and management. Beyond network virtualization technologies, we believe that considering the emergent paradigm of Network Functions Virtualization (NFV) for network-as-a-Service objectives, would be an effective solution for network service management and deployment challenges, and to benefit from NFV network service composition aspects that NFV tend to address. Actually, based on this, NFV is at the intersection between virtualization technologies and X-as-a-Service concepts.

Moreover, network softwarization and programmability concept. This concept relies on the definition of all network behaviors using programmable tools and mainly APIs. Considering this concept goes with the need to go for automation of the deployment, provisioning and management processes within the network. The emerging paradigm that enables network programmability is Software-Defined Networking (SDN) model. SDN allows not only network programmability through APIs but it is also an enabling technology for network virtualization using some of its key features such as planes separation. SDN represents the intersection between network softwarization and network virtualization. Programming the network behavior includes the programmability of network services using Open and dynamic APIs. Therefore, the programmability of network service represents the intersection between network softwarization and X-as-a-Service concept.

1.4 Problem Outline and Research Questions

Network operators are faced with the gap between computing or IT technologies and networking technologies, whereas these present resilient advantages separately.

On one hand, Cloud Computing and Service-Oriented Architecture (SOA) play an important role in the construction and customization of service offerings. Cloud computing relies on the virtualization of computing resources for a dynamic placement of services over virtualized infrastructures. SOA enables a structured interaction of services in heterogeneous distributed systems and presents tools for service description, exposition, composition and orchestration. However, the technologies implementing SOA are not compliant with the conceptual service structure, architecture principles and properties.

On the other hand, emerging networking technologies like SDN and NFV promise to leverage network architecture into more agile network infrastructures. SDN allows network adaptability through network programming. It breaks the vertical integration of network architectures to provide advantages for the automation of network configuration processes and network control. NFV applies virtualization of computing resources to network functions and allows faster deployment cycles of network functions. However, NFV today implements legacy network functions into Virtual Machines without optimization design and keeping the same service logic as defined by standardization bodies. This may be a cost-efficient approach but it does not reduce network complexity and does not eliminate the tight coupling and monolithic nature of network services.

We have focused our research problems over the integration of these technologies as strong enablers for a flexible and dynamic service-oriented network architecture allowing efficient interactions between applications and the network. We have first needed to raise some challenges in each of these areas in order to focus in a next step on the actual convergence. Therefore, within this scope, we have defined three main research questions. Where each research question targets a problem that relies on enablers from two research areas of the scope.

Therefore, the first research question we have defined deals with NFV and XaaS where we have studied the design of virtual network functions as modular elements according to service-oriented models and principles with the objective to provide them as on-demand services in a network platform as-a-Service. This question has also led us to deal with the problem of how service-oriented network functions can be exposed and the impact of using programmbility and dynamic APIs.

The second research question deals with software and virtual network architectures through SDN and NFV to study how these two technologies can be integrated to ensure coherent network control and network management capabilities to have a managed software-defined and virtualized network architecture.

The third research question deals with the challenges of achieving customization and dynamic deployment of network services and virtual networks. It also deals with the problem of how to automate the orchestration or life-cycle management of network services and virtual networks to integrate QoS management and achieve network dynamicity.

1.5 Contributions

The contributions of this PhD thesis can be summarized as follows.

1. Virtual Network Functions Modeling: VNF-as-a-Service

The first contribution is related to the software design of virtual network functions as services. A study of recent component-based modeling and service-based modeling has allowed us to propose a VNF-as-a-Service model for the design of network functions as reusable, composable and mutualizable micro-services. The model proposes a design approach based on functional decomposition, separation of state, and functional independency. It presents a set of development requirements on the structure, the interactions and the management of VNFs. An important aspect of VNFaaS management relies on the separation of concerns into functional and non-functional aspects to introduce an autonomic QoS management. We have applied the VNF-as-a-Service model over the network functions of IP Multimedia Subsystem (IMS) to propose a new IMS-as-a-Service functional architecture with network services composed of VNF-as-a-Service micro-services. This modeling of network functions as services allows their exposition to third parties and composition to build differentiated network services.

2. Network-as-a-Service Architecture

The second contribution is on the architectural dimension as we propose a Network-as-a-Service (NaaS) architecture. The proposed NaaS architecture is a two-layer architecture:

- A Network Exposition Layer that achieves the network openness aspect of NaaS to allow interactions between applications and the network. NaaS exposition Layer includes a QoS-based Description, Discovery, Selection and Composition of VNFs and network services.

- An SDN-enabled NFV Software Network Infrastructure built from the integration of SDN and NFV in order to achieve the networking-level and infrastructure-level adaptability and dynamicity.

The NaaS architecture represents the framework through which VNFaaS and network services are described and exposed. The exposition aims at providing third-parties a mean for expressing their needs in network services. The network provider is then responsible for the deployment of the requested network services.

3. Dynamic Deployment in Network-as-a-Service

The third contribution defines the deployment of VNFs and virtual networks in NaaS. We propose an on-demand dynamic VNF-as-a-Service deployment model and a dynamic deployment model of virtual networks that includes multiple VNFaaS. We then propose an additional virtualization level to consider a “virtual deployment" as a new phase in the life-cycle of a virtual network. The whole virtual network life-cycle management is then automated and performed by a global orchestrator. The objective of the orchestrator is to build and maintain the adequate network service over a dedicated virtual network wile respecting SLA and QoS constraints.

Table 1.1 summarizes the thesis contributions and the publications. These contributions present service and network architectural principles along with deployment models as the required elements to achieve a dynamic Network-as-a-Service architecture for Telcos based on the convergence of new paradigms and technologies.

Contributions	Publication references
Virtual Network Functions Modeling: VNF-as-a-Service	[BOU01] [BOU02] [BOU03] [BOU10]
Network-as-a-Service Architecture	[BOU03] [BOU04] [BOU05] [BOU06] [BOU07] [BOU10]
Dynamic Deployment in Network-as-a-Service	[BOU08] [BOU09] [BOU10]

Table 1.1 – Thesis Contributions

1.6 Manuscript Structure

This PhD thesis manuscript is organized as follows.

In Chapter 2, we deeply study the state-of-the-art for each research domain that defines the thesis scope. In Section 2.2, we analyze service models that consider user-centric approaches to achieve Everything-as-a-Service (XaaS). These include component-oriented model, Service-Oriented Architecture (SOA) and Self-Controlled Components (SCC) and Micro-Services Architecture. We conclude this analysis with a summary and a definition of a set of research challenges. In Section 2.3, we analyze research works on service and network functions virtualization. We study service virtualization in Cloud computing, then virtualization of network functions (NFV) and Cloud networking. We conclude this analysis with a summary and a definition of a set of research challenges. In Section 2.4, we analyze network virtualization and softwarization technologies. We study network service exposition in legacy and NGN networks network virtualization technologies, network slicing technologies and the integration of SDN and NFV technologies. We conclude this analysis with a summary and a definition of a set of research challenges.

In Chapter 3, we present our contributions on Virtual Network Functions (VNF)-as-a-Service Modeling. In Section 3.2, we propose a step-by-step design approach to transform a classic software component into a service component called “as-a-Service" component. It involves structuring, integrating, self-control, designing as-a-Service and describing the component to define a component as-a-Service architecture. In Section 3.3, we propose the software design requirements of as-a-Service modeling. These requirements apply to functional and non-functional design and provide guidelines for the definition of the service structure, service interactions and service management. In Section 3.4, we consider modeling of Telco legacy network functions, we present our proposition for a functional decomposition approach. This decomposition is strongly required to complement the as-a-Service modeling. In Section 3.5, we functionally decompose IMS network functions then model them using as-a-Service design model to propose a new IMS-as-a-Service functional architecture with network services composed of VNF-as-a-Service micro-services. In Section 3.6, we present the implementation of the new IMSaaS architecture with the proposed VNFaaS mirco-services and discuss the obtained evaluation results.

In Chapter 4, we present our contributions for a Network-as-a-Service (NaaS) architecture. In Section 4.2, we define the main actors of NaaS (data, control and management network planes and introduce the features of NaaS (flexibility, dynamicity and adaptability). In Section 4.3, we propose a NaaS Exposition Layer for NaaS architecture where we define service description, service discovery, service selection and service composition.In Section 4.4, we define a NaaS Software-Defined Infrastructure layer based on integrated SDN-enabled NFV. In Section 4.5, we present an implementation feasibility study to validate the two layers of NaaS architecture.

In Chapter 5, we present our contributions for dynamic deployment in NaaS. In Section 5.2, we propose an on-demand and dynamic VNFaaS deployment model. We describe the model in details and define its features. In Section 5.3, we propose a dynamic deployment of virtual networks in NaaS. We define a life-cycle of a virtual network in NaaS, introduce a virtual deployment step for the deployment and define its process, and describe the customization of virtual networks in NaaS. In Section 5.4, we rely on global network service orchestration to achieve the dynamicity of NaaS. We highlight our propositions for the automatic life-cycle management of network services and virtual networks in NaaS. In Section 5.5, we present the implementation and evaluation results of the on-demand dynamic VNFaaS deployment model illustrated with a WebRTC application use-case.

In Chapter 6, we present our conclusions and perspectives where we give a complete overview of the presented contributions, a summary of the contributions in a global view with a projection into 5G, and limitations as research directions that can be considered for future works.

2 State-Of-The-Art

2.1 Introduction

The convergence required to design and build the new network ecosystem becomes possible thanks to the paradigms of “as-a-Service", virtualization and programmability. There is a need to correctly understand their scope and advantages today then accurately refine their use for our objectives. The objectives that we have presented in Chapter 1 have led us to study the state-of-the-art of the research areas defining the thesis scope. Each area is a fundamental aspect to consider when tackling the thesis problem. Therefore, we present in this Chapter a detailed state-of-the-art analysis. First, Section 2.2 studies the evolution of software design models towards Everything-as-a-Service (XaaS). Then, in Section 2.3, we go through the technologies of service virtualization and exposition in Cloud Computing and Network Functions Virtualization in Telco Cloud. Later, in Section 2.4 we present the works related to network exposition, then network virtualization and SDN and NFV integration within network openness perspectives. For each research area, we present the challenges that yet need to be addressed for a Network-as-a-Service objective. Finally, in Section 2.5, we discuss the challenges to conclude this Chapter.

2.2 On Service Models: towards “X-as-a-Service"

Software design models are becoming enablers for efficient Everything-as-a-Service to meet specific needs by providing customized services. We present in this Section, the chronological evolution of patterns issued from Component-Based Software Engineering (CBSE) [1]. We survey architectural models from component-oriented models (Section 2.2.1) and service-oriented model (Section 2.2.2) to Self-Controlled Component model (Section 2.2.3) and Micro-Services (Section 2.2.4). We highlight and analyze the strengths and shortcomings of each model as well as the associated management approaches. We study their relevance and extract further requirements to be integrated in an "as-a-Service" architectural model.

2.2.1 Component-Oriented Models

In CBSE, component models offer a structured programming paradigm for building applications allowing to reuse software components. A software component is a software module or unit offering a service and able to communicate with other components. A component has a structure that represents the architectural unit. The functionality that a component provides is defined with its dependencies using offered and requested interfaces in a structured way, usually hierarchical, form where components can either be primitives or composites (i.e., containing one or many inner components). This concept is adapted for the specification of large-scale distributed systems. Let us analyze the architecture of a component in different initiatives. We focus on models that targeted user-centric services.

Fractal Component Model

Fractal [2] [3] [4] defines a general conceptual component model. It is defined in OW2 Consortium as a modular and extensible component model that can be used to design, implement, deploy, reconfigure and manage complex software systems and applications, from operating systems and middle-ware platforms to graphical user interfaces. *Fractale* stands for objects that have a regular invariant structure even at different scales. This scalability and elasticity feature of Fractal is important for distributed systems. In Fractal, dynamicity aspects are presented through the possibility to add or remove components to or from an already deployed application thanks to reconfiguration of the bindings between components (at run-time scaling) and to the introspection of composite components. Figure 2.1 shows the architecture of a system of Fractal components. Server interfaces, client interfaces and controllers are respectively represented in red, green and blue. The strength of Fractal resides in its interfaces:

- Usage interfaces: client and server interfaces for functional bindings, where the component defines what it needs and provides,
- Control and management interfaces: they represent controllers also called *Membrane* responsible for the non-functional aspects. These interfaces are used to control and manage the component life cycle and provides methods to configure, start and stop the component.

In addition to that, Fractal defines an Architecture Description Language (ADL). The ADL uses an XML syntax and is a way to describe a component based system without having to worry about the implementation code.

However, Fractal component server interfaces may be functional interfaces or non-functional interfaces (controllers or membrane). The functional connections are well-defined and composition is possible, but the non-functional interfaces (the membrane) are not structured. Indeed, control and management are not separated. To address these shortcomings, the Grid Component Model (GCM) was proposed as we explain next.

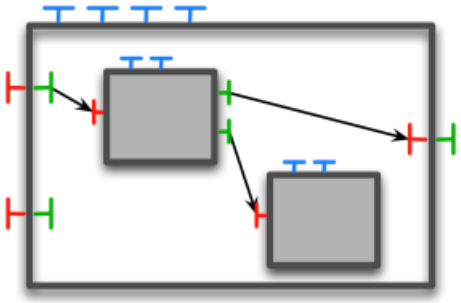

Figure 2.1 – Fractal Component Architecture

Grid Component Model (GCM)

Grid Component Model (GCM) [5] has extended Fractal towards large-scale autonomous distributed systems. It is specified by ETSI [6]. In addition to Fractal advantages and reconfiguration capacity, GCM brought collective communication interfaces for multicast, unicast and gathercast between components. Moreover, it defines well structured non-functional aspects through non-functional components in the membrane that have their own functions and defined with all necessary management elements and interfaces. Figure 2.2 shows the architecture of a GCM component with non-functional components in the membrane.

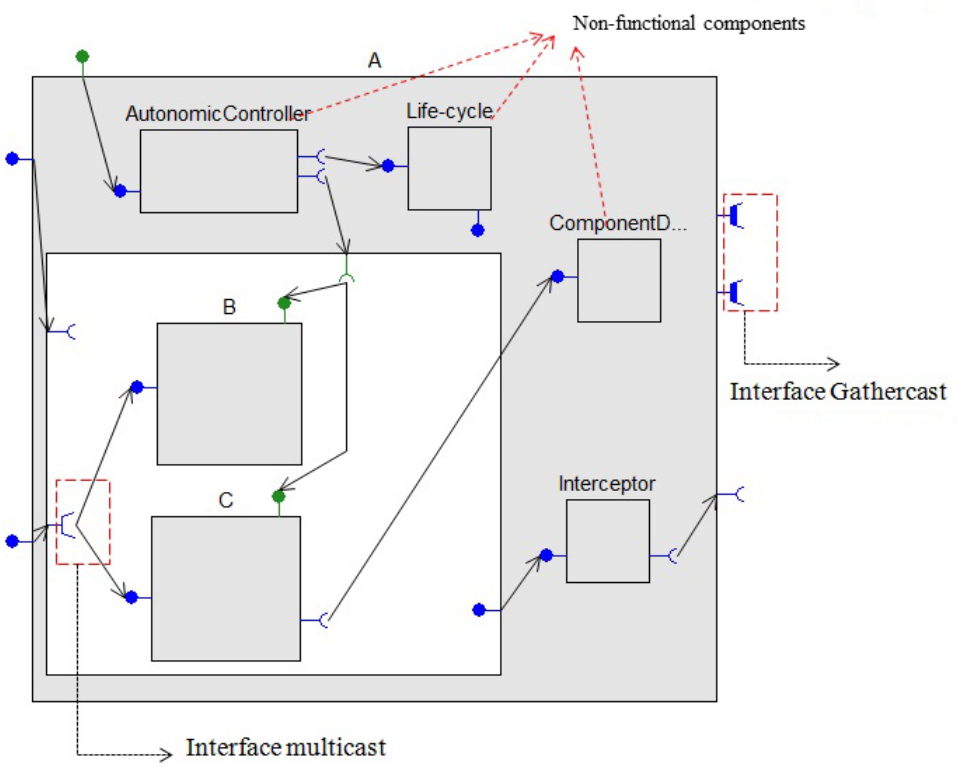

Figure 2.2 – GCM Component Architecture

In GCM, the membrane structure is a strong asset for component management and control but also for non-functional bindings or compositions which become possible similarly to the functional bindings in Fractal. For the functional aspects, communication is performed on interfaces and follows Fractal component bindings. Furthermore, GCM architecture separates concerns as it separately designs the Business Content and the Membrane. The Business content represents functional aspects of a primitive component responsible for business service logic. The structured Membrane is responsible for non-functional aspects holding the management functions. The structure of GCM components allws management using autonomic approaches in hierarchical architectures by implementing Monitoring-Analysis-Planning-Execution (MAPE) [7] loop to manage this hierarchy where *Execution* is used to change the bindings. Figure 2.3 shows a GCM component architecture with a MAPE loop.

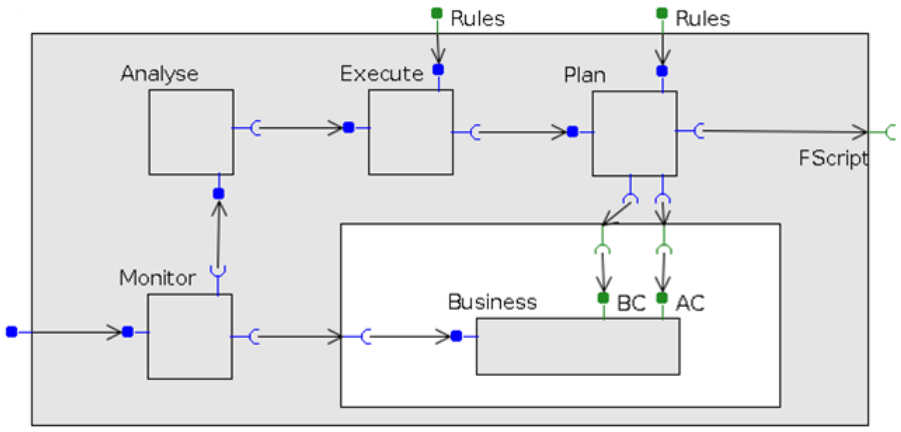

Figure 2.3 – GCM Component Architecture with autonomic control (MAPE loop)

However, the main shortcoming of GCM is this hierarchical nature of the bindings between components which adds functional coupling and makes primitive components hard to be composed and recomposed. Bindings are needed to be less tight in order to have a dynamic composition. Service-Oriented Architecture (SOA) has addressed this rigidity in bindings as we present hereafter.

2.2.2 Service-Oriented Architecture (SOA)

Service-Oriented Architecture (SOA) [8] [9] is an architectural model where a component is a *service*. It is the unit of work of a service provider to achieve a desired service for a service consumer. It proposes a composition model with loose coupling and reduced functional coupling among interacting service components. These features are presented as a must for building customized services. Composition consists in building a global service composed of a set of elementary service components. Composition [10] is customizable by adding, replacing and removing service components based on user needs. SOA proposes a structure: Service Component Architecture (SCA), and a technology: Web Services (WS).

Service Component Architecture (SCA)

SCA [11] is a component model adapted to SOA properties and enables creation of service components and modeling of service compositions. The strong properties of SCA are: reusability, reference points and loose coupling. Reference points allow the service component to have all the needed connections, while loose coupling brings flexible compositions of service components through loose bindings to eliminate all functional couplings. The coordination of service components is achieved through Business Process Execution Language (BPEL). Figure 2.4 shows the architecture of a service component.

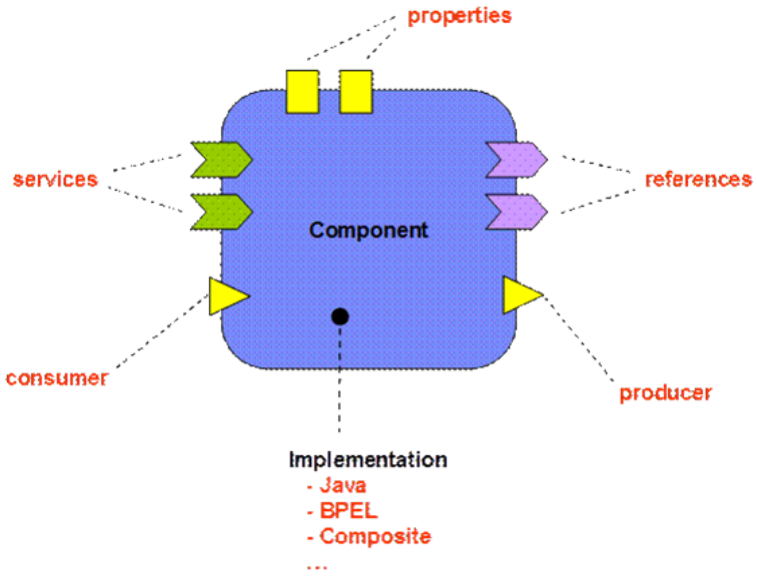

Figure 2.4 – Service Component Architecture

Web Services

Web services [12] implement SOA requirements. They thus support flexible composition and methods of service integration in Cloud applications. The strong asset of web services is the Client-Provider relationship based on service publication and description in catalogs by the service provider, and service discovery and invocation by the client. For this, they support Web Services Description Language (WSDL), Universal Description Discovery and Integration (UDDI), EXtensible Markup Language (XML) and APIs such as Simple Object Access Protocol (SOAP) and Representational State Transfer (REST). However, actually, coupling in SCA is not loose in practice. Indeed, the reusability feature in BPEL is strong and relies on existent functional coupling. Thus, SCA model is not dynamic and it is not be possible to add, replace or remove service components or bindings dynamically at run-time. Also, web services do not support QoS management and thus cannot integrate the control of Service Level Agreement (SLA) compliance. Furthermore, SOA does not consider service distribution while GCM does.

2.2.3 Self-Controlled service Component Model (SCC)

Self-Controlled service Component Model (SCC) [13] combines the strong features of GCM and SOA. SCC services are designed based on the resilient GCM membrane structure with explicit non-functional compositions and autonomic management; and on loose coupling, highly dynamic composition and service discovery of SOA concepts. SCC has adopted GCM and at a certain extent MAPE loop. SCC has designed components based on initial SOA features to inherit the ability to achieve dynamic at run-time service composition. Further, SCC model has defined new service component features beyond the ones advocated by existing models. It has proposed a dynamicity in handling components through auto-control approach for the management of non-functional aspects (QoS) in flat architectures enabled by the membrane structure. Moreover, SCC components include functional and non-functional specifications starting from the design step. Furthermore, self-control mechanisms are attached automatically to SCCs to enable autonomic management at run-time (Figure 2.5).

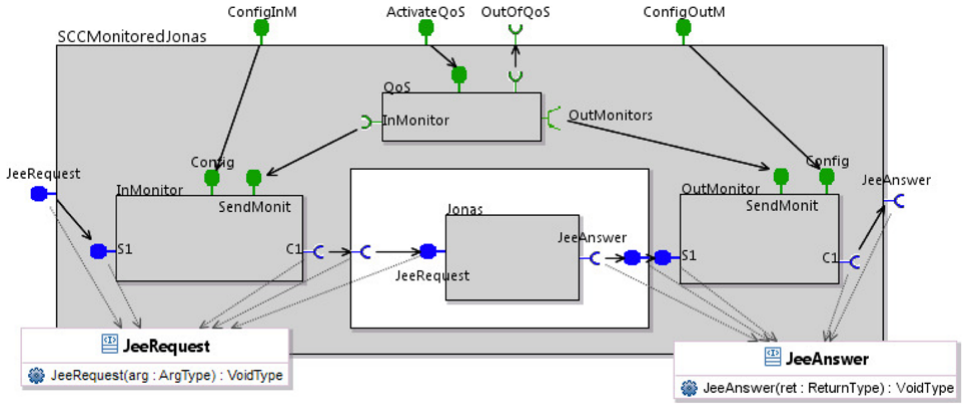

Figure 2.5 – Structure of a Self-Controlled Component

2.2.4 Micro-services Architecture

Regarding the realization of services, Micro-services Architecture [14] [15], has been proposed to cope with deployment and scalability issues of large-scale monolithic applications and services [16]. A micro-service is defined as an independent service component or process that implements only functionalities that belong to its basic concern and is designed independently from any implementation specifications. Therefore, a micro-service architecture is a distributed application whose components are micro-services. As they are independent components, there should be, conceptual, deployment isolation. Since all the components of a microservice architecture are microservices, its behaviour results from the composition and coordination of its components. The microservice architecture does not specify any particular programming paradigm. It provides guidelines to decompose the components of a distributed application into independent entities, each addressing a concern [17] [18]. With the micro-service architecture, the objective is to have a realization of services as close as possible from the conceptual design of the application or service architecture.

2.2.5 Summary and Challenges

We have analyzed in this Section existing models within the “as-a-Service" area and we present here the related challenges. This analysis has allowed us to study the principles and evaluate the relevance through the strengths and shortcomings of component and service models that targeted user-centric services in virtualized and distributed environments.

Our objective behind this analysis was to define the most accurate model and principles for adoption within network services and particularly for network functions in Telco NFV environments.

In order to define such principles, the shortcomings of existing models need first be addressed. In component-based models, functional dependencies are explicitly defined for deployment and compositions are static because of tight coupling. Thus, instantiation of components follows an explicit service request that cannot be modified. Further, the non-functional behavioral aspects, i.e., regarding the QoS are ignored.

In service-oriented models, publication and discovery of services is possible but services do not apply SOA properties and mainly statelessness. Further, the SCA structure allows a hierarchical and structural composition but does not allow a dynamic reconfiguration of services. The QoS aspects are here also ignored. For these shortcomings enablers exist though.

Indeed, the service-oriented features of SCC through the functional and non-functional specifications and properties represent a strong basis for an accurate service model for network services specifications. It takes into account the behavioral aspects and their management. Moreover, the micro-services architecture pace the way for eliminating the monolithic nature of services and is adapted for large-scale distributed services, which is the case of Telco network functions and NFV services.

Therefore, based on the stakes and awaited objectives regarding the adoption of component models for the design of network function in Telco virtualized environments, SCC model and micro-services architecture are resilient enablers to adopt.

But, with the objective of building differentiated and customized network services for users' needs based on at-run-time dynamic compositions of loosely coupled network functions, what features do we need to retain for network functions? How to apply and extend these service models with their architectures and principles to define a model for network functions? How to design network functions to make them become “as-a-Service" with all required properties?

Let us present next, virtualization technologies and how service models may impact them.

2.3 On Service and Network Function Virtualization

Virtualization provides a mechanism to decouple the applications from the hardware support necessary for their operation. This capability enables Data Centers to support diverse applications using a limited number of physical servers. The mechanism allowing this decoupling is the Virtual Machine (VM) that defines an application and its associated functions. In Cloud environments, computing and storage hardware virtualization allows to define virtual machines (VMs) where cloud applications are deployed. This allows to effectively manage Cloud services. This way, hardware virtualization contributes mainly in the optimization of hardware resources use in the Cloud. In addition, like IT, network operators have also been interested in “cloudifying" the network through Network Functions Virtualization (NFV). The objective is to achieve dynamic deployments of network functions and efficient network service management. Furthermore, virtualization of network links has played a key role in virtualized environments, commonly known as Cloud Networking.

Therefore, in this Section, we analyze the virtualization and deployment process in different virtualized environments. In Section 2.3.1, we consider service exposition and deployment in the Cloud. We then consider network functions virtualization and deployment in NFV Telco Cloud in Section 2.3.2. Later, we highlight the definition of Cloud Networking in Section 2.3.3. Finally, we present the challenges that need to be raised for each environment in Section 2.3.4.

2.3.1 Service Virtualization in Cloud Computing

One of the most significant recent advances in the field of information technology is Cloud computing. It is the technology that applied service-orientation (SOA) principles for service conception, service composition and service delivery aspects. Cloud computing is a large scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically scalable computing functions and services are delivered on demand to external customers over the Internet [19]. A technical foundation of Cloud computing lies in the virtualization of various computing resources, which is essentially an abstraction of logical functions from underlying physical resources. Cloud services represent remote delivery of computing resources, most often via the Internet. This is especially relevant in public Cloud environments where customers obtain services from a third-party Cloud provider. From a service provisioning perspective, Cloud services consist of not only computing functions provided by a Cloud infrastructure but also communications functions offered by the Internet.

According to the National Institute of Standards and Technology (NIST), Cloud computing [20] is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This Cloud model is composed of three service models and five essential characteristics.

Cloud Service Models

Figure 2.6 shows the SOA-Cloud architecture model. We distinguish three layers as defined by NIST [20]: Infrastructure-as-a-Service, Platform-as-a-Service and Software-as-a-Service.

Infrastructure as a Service (IaaS): The service provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components.

Platform as a Service (PaaS): The service provided to the consumer is to deploy consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure, but has control over the deployed applications and configuration settings for the application-hosting environment.

Software as a Service (SaaS): The service provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface or a program interface. The consumer does not manage or control the individual application capabilities, but can perform user-specific application configuration settings for customization.

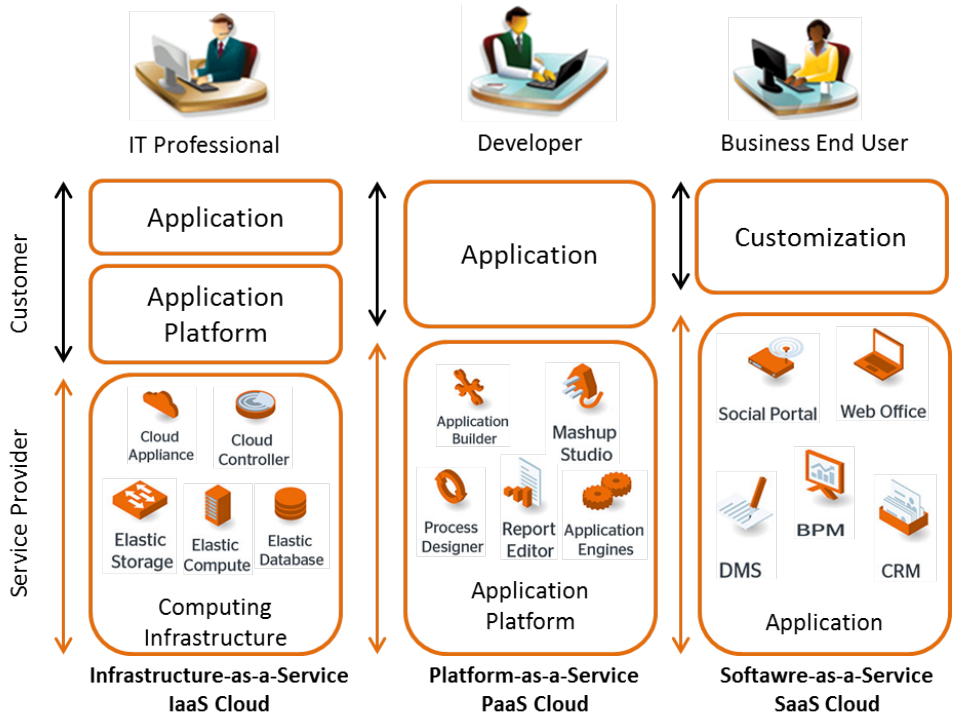

Figure 2.6 – Cloud Architecture Model

Cloud Computing Characteristics

Cloud computing characteristics can be seen from a service provider perspective and from a client perspective. Cloud Computing characteristics include:

On-demand self-service: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

Broad access: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous client platforms (e.g., mobile phones, tablets, laptops,...).

Resource pooling: The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data-center). Examples of resources include storage, processing, memory, and network bandwidth.

Rapid elasticity: Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

Measured service: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer.

Cloud service models rely on SOA-based exposition as we explain hereafter.

Service Exposition in Cloud Computing

Exposition approaches and models have been driven by the adoption of service models, mainly SOA, as an enabler for virtualization in Cloud.

Indeed, exposition aims at abstracting services from the Cloud and network infrastructures to make their realization transparent for users. In Cloud computing, SOA essentially facilitates virtualization of computing infrastructures by encapsulating resources and capabilities in the form of services and provides loose coupling interactions between them. We present in this paragraph, a survey on SOA-based service exposition in Cloud Computing.

The SOA principle and its implementation technologies have become state of the art of information service delivery and have been widely applied in various distributed computing areas and mainly Cloud computing. It provides effective architectural principles for heterogeneous system integration. Though SOA can be implemented with different technologies, Web services provide a preferred environment for realizing SOA as we have presented in Section 2.2.2. A Web service has an interface that describes a collection of operations that are network accessible through standardized XML messaging [21]. A Web service is described using a standard, formal XML notion, called its service description. It covers all the details necessary to interact with the service, including message formats, transport protocols, and location. The interface hides the implementation details of the service, allowing it to be used independently of its implementation. This enables Web services-based systems to be loosely coupled, component-oriented, with cross-technology implementations. Key elements of a Web service-based implementation of SOA include service provider, service broker/registry, and service customer. The basic operations involved in the interaction among these elements are service description publication, service discovery, and service binding which is the concrete access to the service. Service composition is also an important operation for customizing services based on users' requirements.

Representational State Transfer (REST) can be considered as an alternative to the standard Web service technologies for implementing SOA systems [22]. In REST, the focus is on the resources exposed by services. Each resource is identified by a URI represented by a certain MIME type (such as XML or JSON), and accessed and controlled using POST, GET, PUT, or DELETE http methods. SOA services based on REST expose the resources that they manage through (and only through) these four methods. This set of technologies that follow the REST design style for realizing service-orientation is conventionally called RESTful Web services.

Serving as a key enabler to virtualization, SOA forms a core element in the technical foundation of Cloud computing. The advantages of SOA and virtualization in the Cloud computing ecosystem have been widely described [23] [24] [25]. SOA has been adopted as the main model for Cloud service provisioning. Following this model, various virtualized computing resources, including both hardware (e.g. CPU capacity and storage space) and software applications are delivered to customers as services through the IaaS, PaaS and SaaS paradigms. The Open Grid Forum (OGF) works on the Open Cloud Computing Interface (OCCI) standard [26] to define SOA-compliant open interfaces for interacting with Cloud infrastructures. The Cloud service provisioning is strongly influenced by SOA principle. Cloud services for VMs like Amazon Elastic Cloud Computing (EC2) [27] and services for plain storage like Simple Storage Service (S3) [28] expose their services via Web service interfaces. Alternatively, GoGrid [29] IaaS defines its interfaces to expose control services of hardware resources in Cloud infrastructure using REST technologies as an SOA implementation instead of SOA web services.

The exposition of services in Cloud cannot be decoupled from deployment. The selected services are then deployed over a Cloud infrastructure. Let us then study in the next paragraph service deployment in Cloud Computing.

Service Deployment in Cloud Computing

A VM defines resources to be associated at the implementation phase. The deployment of services in the context of Cloud Computing consists in generating machine images then instantiating them in the form of VMs. These VMs are targeted to ensure the proper execution of a cloud application over the IaaS. The hosting infrastructure provider offers the requested physical resources in an Infrastructure as-a-Service (IaaS) model [30]. Different virtualization technologies may be used at the IaaS level [31]: Xen [32], KVM [33] and VMware [34] allowing to offer computing resources on demand to respond to requests of VMs creation, migration and scaling. Different cloud infrastructure vendors provide IT resources on-demand like Amazon EC2 [27] and Google Compute Engine [35].

However, local aspects of configuration and aspects related to VMs interconnection are to be resolved during the deployment phase. This means that some operations at the placement phase are yet to be addressed. So how to place a maximum of VMs optimally in a minimum number of servers to optimize resource utilization across the physical infrastructure while maintaining the quality required for the application operation?

Actually, the placement of VMs is treated in the literature through different techniques of decision making [36] [37]: reservations [27], on-demand access [38] and spot markets [39]. As we can see, placement is in all cases guided by physical constraints regarding the utilization of hardware resources in the infrastructure. This is the case for VMware [40] and OpenStack [41] which both follow this model. Computing and storage resources (CPU, disk space and RAM) are the only conditions to satisfy when placing VMs.

Most VMs placement policies focus mainly on the effectiveness and efficiency of computing and storage resources utilization while network aspects are widely ignored and limited to local connectivity aspects. To avoid network problems, solutions tend to place VMs in the same cluster so the communication between VMs would be local to a cluster switch. But, what about the deployment of services considering the network environment at a larger scale?

Let us see next whether this question is addressed in today's cloudified network through NFV.

2.3.2 Network Functions Virtualization (NFV)

Virtualization has had a major impact in the computing world, and is now impacting the networking world as well. NFV is an initiative to move the network functions and services that are now being carried out by proprietary, closed, dedicated hardware to virtual containers in a form of Virtual Network Functions (VNFs) on Commercial-Off-The-Shelf (COTS) hardware. It promises a decrease in the amount of proprietary hardware that is needed to launch and operate network services within the operator network infrastructure. Therefore, it should overcome network complexity and facilitates network management.

NFV main benefit is the dynamic deployment of network services over a virtualized infrastructure. It then reduces service deployment cycles and thus Time-To-Market. In ETSI NFV architecture [42] [43] (Figure 2.7), the NFV Infrastructure (NFVI) is virtualized and comprises a set of virtual resources providing computing, storage and network facilities. NFVI resources are controlled and managed by the Virtualized Infrastructure Manager (VIM). At the intermediate level, VNFs run on top of one or several NFVI resources and the VNF Manager (VNFM) and Element Management System (EMS) are responsible for controlling and managing VNF resources. NFV handles VNF lifecycles (on-boarding, instantiating, monitoring, terminating and deleting). An NFV service is built based on a combination of VNFs using VNF Forwarding Graphs (VNF FGs). The NFV Orchestrator (NFVO) controls and manages infrastructure services. It coordinates the resource allocation to infrastructure services and VNFs, either by a direct interaction with the VIM or via the VNF Manager. It takes into account deployment policies based on various criteria including affinity rules, location constraints and performance criteria. The orchestrator brings more intelligence to the whole system comparing to the operations performed by the OSS/BSS which perform traditional management operations. NFV covers mainly service deployment issues which makes it a resilient opportunity for Telcos.

Like IT, Telcos has worked to “cloudify" the network through NFV in order to be able to implement network functions logics in VMs over a virtualized infrastructure in the same way as in a Cloud Computing environment. This is why NFV is also called Telco or operator Cloud. When applying the Cloud model at a network level (network cloudification) by analogy and in a symetric manner, different service models are considered in network service delivery.

Network Functions Virtualization Service Models

NFV has also defined the following three service models [44]:

Virtual Network Function as a Service (VNFaaS): provides access to one or a set of VNFs instantiated on Points of Presence in NFV Infrastructure. This model represents a service in Network Software-as-a-Service layer and provides, similarly to Cloud SaaS, turnkey services. The difference is that NSaaS services belong to the network service delivery level.

Virtual Network Platform as a Service (VNPaaS): provides a platform to provide VNFs with programming tools and virtual links. This model represents services of the Network Platform-as-a-Service (NPaaS) layer, somehow equivalent to Cloud PaaS, and proposes a service platform with catalogs of exposed services. It thus allows users to choose and compose VNFs such as AAA, NAT, and firewalls, but also functions like routing and addressing.

Network Function Virtualization Infrastructure as a Service (NFVIaaS): the NFVI, as its name implies, consists in the virtualized physical network infrastructure. In a similar model as Cloud IaaS, this model offers execution environments for VNFs i.e., computing and storage capabilities but also networking capabilities for functions like routing and switching. It is thus considered as the Network Infrastructure as-a-Service (NIaaS) layer.

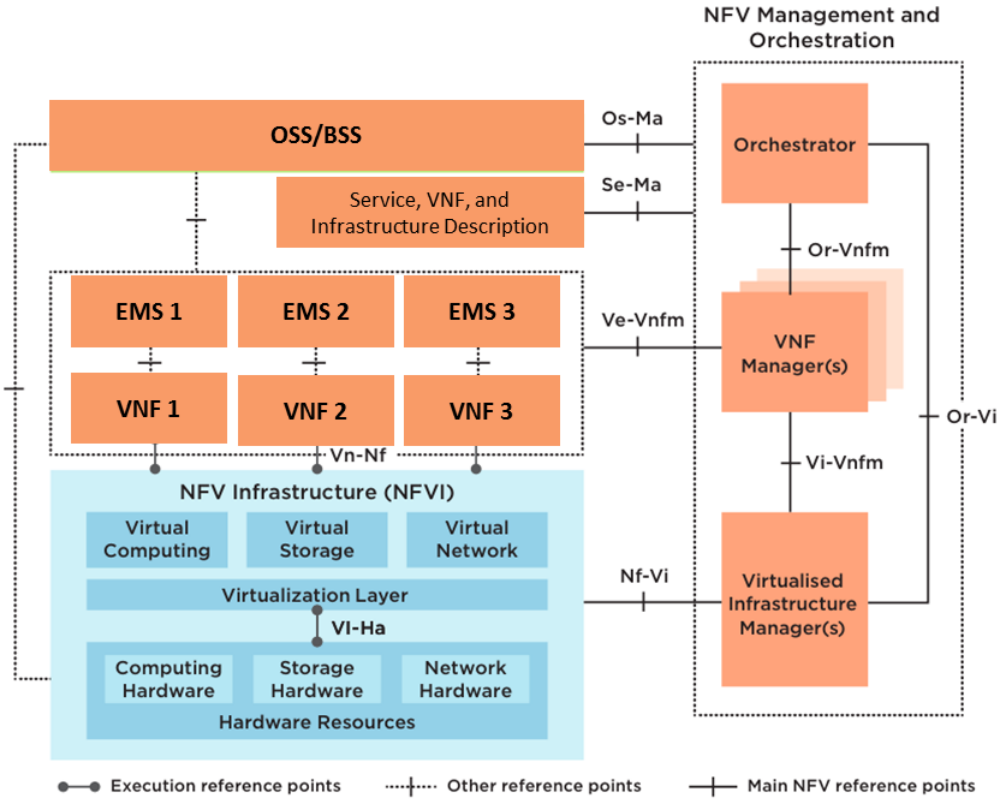

Figure 2.7 – NFV Architecture Framework

Deployment of Virtual Network Functions in NFV

To go in the direction of as-a-Service virtualization, ETSI NFV initiative considers cloudification of operator networks through the virtualization of the physical network functions in legacy networks, through different use-cases [44]. These include the virtualiztion of IP Multi-media Subsystem (IMS) which is a session control architecture that supports the provisioning of multimedia services over Evolved Packet Core (EPC) and IP-based networks, virtualization of EPC which is the latest core network architecture for mobile networks, and virtualization of Content Distribution Networks (CDN),...etc. Network functions in mobile and converged networks are defined based on 3GPP standards and specifications. In IMS, network functions include: P-CSCF, S-CSCF, I-CSCF, MGCF and HSS. In EPC, examples of network functions include: MME, S/PGW and PCRF. In CDN, network functions are the main CDN components: CDN Controllers and cache nodes.

Different actors have worked to present implementations of these NFV use-cases. They have considered the virtualization of IMS, EPC and CDN using either approaches that focus on very specific entities (for example HSS, presence services and Application Servers in IMS), or approaches that deal with the architecture as a whole to treat the general problem of network and cloud integration.

VNFs are the central elements within each of NFV service models where VNF Management functions perform life-cycle management based on VNFs' requirements. At instantiation, NFVI resources are assigned to a VNF based on these requirements, but also taking into consideration specific constraints and policies that have been pre-provisioned or accompanying the request for instantiation [45]. During the life-cycle of a VNF, the VNF Management functions may monitor KPIs of a VNF and may use this information for scaling operations. Scaling may include changing the configuration of the virtualized resources: scale up (add CPU), scale down (remove CPU); adding new virtualized resources: scale out (add a new VM), shutting down and removing VM instances (scale in), or releasing some virtualized resources (scale down).

Therefore, the deployment of VNFs in NFV does consider only the requirements related to the computing resources needed for the execution of the VMs that contains the VNFs. So the deployment process in NFV, as well, focuses primarily on the effectiveness and efficiency of computing resources utilization. We believe that this way of implementing VNFs is reducing the benefits of NFV that Telcos may take advantage of. Furthermore, NFV is a strong enabler necessary for the definition of a resilient virtual network deployment process. But is it alone enough in the definition of such a deployment?

Before highlighting the challenges related to NFV today, let us analyze how networks in virtualized environments are defined.

2.3.3 Cloud Networking

It is first important to recall that Cloud Networking has emerged from the need to interconnect not only locally via traditional Virtual Local Area Networks (VLANs), but through the WAN, distant and distributed data centers or elements of a data center such as servers. Cloud Networking started to take shape with the consideration of QoS constraints for applications and cloud services that are hosted in VMs within the Cloud. But from a network point of view, VM placement and operation are not directed only by network links capabilities.

In fact, the links and interconnections between VMs have a strong impact over distributed application performance and over the respect of the required QoS level. These problems have recently been studied in [46] and [47]. These works are based on distributed Clouds whose problems concern allocation of geographically distributed resources for which interconnection networks are needed. They propose algorithms for resource allocation in distributed data center. The work in [46] presents a study on the placement of VMs hosting applications and the migration of VM in case of increased latency or congestion over the network. The proposed algorithm is based on minimizing time consumption data transfer. It focuses on VMs placement policy taking into account network requirements. The proposed VM migration policy considered unstable network connections which impacts application performance.

In the proposed VMs placement policy, the approach seeks to optimize access to data by placing the VM on the physical machine with the shortest data transfer time. In the same direction, the work in [47] proposes to reduce the distance and latency between selected data centers by minimizing traffic inter and intra data center. It also seeks to minimize data paths and to maintain applications performance. Several steps are involved in the allocation of resources: selection of data centers, VMs partitioning over selected data centers, identification of existing physical resources in each data center and VMs placement.

By analyzing these works, we can say that in cloud Networking, the links defined by today's network function chaining mechanisms remain forwarding links defined after VMs placement. Consideration of application requirements at the transport network level before or when deploying and placing VMs or VNFs is not yet considered in Cloud Networking and in NFV. We will analyze further Service Function Chaining in SDN and NFV contexts in Section 2.4.

At this stage, the analysis of virtualization of infrastructures in Cloud Computing and the virtualization of network infrastructures and network functions in NFV, as well as the virtualization of networking links in Cloud has led us to a set of challenges as we present next.

2.3.4 Summary and Challenges

In today's user-centric ecosystem, network providers are working to take advantage from NFV to deliver network services with high dynamicity to meet the continuously changing network users' requests. However, adoption of NFV by network operators is still at a first step.

Indeed, NFV implementations are Cloud-based but do not integrate all Cloud characteristics. They correspond only to an Infrastructure-as-a-Service (IaaS) model. VNFs are VMs corresponding exactly to the network functions that network operators have in legacy network architectures. Basically, the operation of VNF deployment corresponds to a VM placement over the NFVI. Thus, it is clear that to date, NFV consists in implementing legacy network functions logic, performed today by physical proprietary dedicated hardware, in software systems over VMs of the NFVI keeping the same service logic as defined by 3GPP specifications. Actually, this way of implementing network functions logic in VMs over virtualized infrastructures is so similar to implementing applications in VMs in a Cloud Computing environment. So, NFV today is virtualization of network equipment, a way to "cloudify" the operator network by virtualizing legacy physical network functions. Simply virtualizing existing network functions (e.g., gateways, CSCFs) may make them cheaper to implement but will not reduce the network complexity and does not eliminate the monolithic nature of network services and architectures. How can we push NFV to obtain Cloud characteristics for networks?

Moreover, NFV promises the ability to provide tailored network services based on compositions of VNFs offered as services similarly to cloud services. However, Cloud services are based on well-defined Service and Component based Models like SOA. So, we strongly believe that NFV should apply adapted CBSE service models principles for the software design of VNFs.

The design of network functions is crutial to leverage the benefits of NFV. There are several enablers for that such as Network-to-Cloud convergence through service-orientation [48] and micro-services architectures.

Furthermore, regarding the targeted network service customization, research directions for NFV include reconsidering functional architectures and the granularity levels of network functions. That means to separate the functional entities' logic and data or state, to consider decomposing the network functional entities' logic into smaller sub-functional entities, which should lead to finer control over the distinct functions. These are major research directions we will consider in our work. We believe that for greater flexibility and to be able to offer differentiated network services, network operators should reconsider their legacy functional architectures to rethink the services' logic to make them less monolithic. The challenge here is to push further the actual VNF developer/provider decomposition into a functional decomposition of VNFs. So, apart from evident benefits like resource utilization that NFV surely brings through this cloudification of network functions, how can we take more advantage of advances in Cloud computing and Cloud application architectures within this network-to-Cloud convergence to find the best way to architect VNFs to provide more flexible NFV network service offerings?

Regarding Cloud Networking, the links defined by today's service chaining mechanisms remain forwarding links defined after VMs placement. Consideration of application requirements at the transport network level before or when deploying and placing VNFs or VMs is not yet considered neither in NFV or in Cloud Networking. Indeed, considering links and NFV chaining in the definition of network virtualization is different than considering each virtual network as a chaining of network applications or network functions in a customized manner.

In existing service and network virtualization approaches, the visibility considered at the deployment process is either a placement of virtual machines or VNFs over commodity hardware with computing resource optimization (the case of Cloud Computing and NFV) or an optimization of physical links (the case of Cloud networking). We have come to the conclusion that in today's virtualized environments, the deployment is actually a “placement" which addresses a different problem comparing to the notion of “deployment". None of the approaches analyzed above treats the notion of end-to-end network flow which is intended to support applications constraints and needs from the network. This limitation has motivated our reflections for the design and the deployment of network functions which need to be in line with the “user-centric" behavior of a user in mobility and having preferences linked to his location. So how to customize network services and virtual networks for such applications and keep them ready to change dynamically as application requirements change?

Virtualization has been the enabler for deep changes in Cloud and Telco environments. SOA was a key enabler to exposition and virtualization in Cloud. Therefore, let us study in the following Section, research works related to network exposition then the evolution towards SDN-based and NFV-based architectures and their likeliness to enable network exposition.

2.4 On Network Virtualization and Softwarization

Software-Defined Networks (SDN) [49] and Network Function Virtualization (NFV) [50] have emerged as key technologies to transform Telco networks. They promise to address flexibility in network service delivery. SDN technology gains in maturity and NFV becomes a real technological trend. NFV allows flexible service deployment and hardware independency as it targets the placement of software-defined functions over virtualized infrastructures. However, NFV does not specify the networking logics between the functions to deploy. Unlike SDN, it has no knowledge of the network state. Based on the centralized network control, SDN allows efficient network policies enforcement using network abstractions and programmability. It provides automated network control and management and a dynamic realization of data plane functions. But SDN does not indicate the deployment of these functions over VMs, while NFV does. This complementarity is full of promise for an implementation of SDN and NFV within a same network framework. Therefore, beyond SDN and NFV inherent benefits separately, the evolution towards softwarized and virtualized networks where services can be exposed and delivered dynamically in an as-a-Service manner calls for the integration of SDN and NFV features. This also involves dynamic service chaining challenges. Moreover, SDN and NFV contribute to redefining network virtualization. The concept of Slicing is thus emerging through SDN-based and NFV-based network virtualization solutions.

We survey in this Section, research works related to network exposition in legacy architectures before the emergence of SDN and NFV (Section 2.4.1). Then, in Sections 2.4.2 and 2.4.3, we analyze existing network virtualization and network slicing solutions. Later, with the evolution of Telco networks through SDN and NFV, we focus in Section 2.4.4 on the state-of-the-art regarding the integration of SDN and NFV. Finally, in Section 2.4.5, we highlight challenges regarding the evolution of Telco networks towards flexible and on-demand service delivery.

2.4.1 Network Service Exposition

In networks, SOA supports virtualization of networking resources in the form of SOA network services, commonly known as “service-oriented network virtualization" and is a strong enabler for the NaaS paradigm. It allows network infrastructures to be exposed and utilized as network services. Therefore, SOA greatly contributes to a convergence of Cloud and networks to realize the NaaS paradigm. In the telecommunication domain, the idea of separating service-related functions from data transport mechanisms was motivated by the need to create new market-driven applications by reusing existing service components. This separation allows underlying network infrastructure to be virtualized and shared by service-related functions in order to create various applications.

Efforts toward making telecom network a programmable environment for delivering value-added services go back to Intelligent Network (IN) [51]. IN defined overlay service architecture on top of physical network infrastructure and extract service intelligence into dedicated service control points. IN was driven by Remote Procedure Call and functional programming.

Some Telecom API standards, like Open Service Architecture (OSA) Parlay framework and Java API for Integrated Networks (JAIN) [52], were developed for achieving a similar objective as IN but with easier service development. These APIs simplify telecom service development by abstracting signaling protocol details of the underlying networks. Parlay/OSA and JAIN were implemented based on Common Object Request Broker Architecture (CORBA) and Java Remote-Method Invocation (RMI) technologies. System modules in such distributed computing technologies are essentially tightly coupled. This hinders a full support for networking resource abstraction and virtualization. Based on the emergence of Web service technologies, Parlay X was developed by the Parlay Group, ETSI, and 3GPP [53]. The objective of Parlay X is to offer a higher level of abstraction than Parlay/OSA in order to allow developers to design and build value-added telecom applications without knowing details of networking protocols. Web services technologies are employed in Parlay X to expose networking capabilities to upper layer applications, which opens a door for applying SOA to realize the separation of service provisioning and data transportation.

Later, Telecom systems transition toward multi-service IP-based networks has resulted in two main developments : Next Generation Network (NGN) [51] and IP-based Multimedia Subsystem (IMS) [54]. NGN is defined by ITU-T as a packet-based network able to provide services and make use of multiple broadband QoS-enabled transport technologies where service related functions are independent from underlying transport technologies. IMS is defined by 3GPP and ETSI to realize the NGN concept. The key feature of the NGN architecture is the decoupling of network transport and service-related functions, which allows virtualization of network infrastructure for flexible service provisioning.

Tight coupling between service provisioning and network infrastructure became fast a barrier to rapid and flexible service development and deployment. The problem of silo mode of telecom service provisioning has been studied in research to build a Service Delivery Platform (SDP). Main SDP specifications include OMA Open Service Environment (OSE) [55] and TM Forum Service Delivery Framework (SDF) [56]. The core idea is to have a framework for service management and operation by aggregating the network capabilities and service management functions in a common platform. The objective of SDP is to provide an environment in which upper layer applications can be easily developed by combining underlying networking capabilities and also enable collaboration across network service providers, content providers, and third-party service providers. The web services approach has become a de facto standard for communications among system components in SDP. Web service orchestration technologies such as BPEL [57] are also becoming part of SDP for enabling services to be composed with both telecom functional blocks and business logic/applications in the computing domain.

In front of the objective to allow network and computing service providers, content providers, and end-users to offer and consume network services, there was a huge need for an efficient way of service and application delivery, that is also user-centric. This is a challenge that has not been sufficiently addressed by the works mentioned above such as IMS, NGN, and SDP.

Therefore, organizing the services offered by various networks on an overlay that allows service providers to offer rich services became the new objective. For that, IEEE developed the Next Generation Service Overlay Network (NGSON) standard [58] [59]. It specifies context-aware, dynamically adaptive, and self-organizing networking capabilities including both service level and transport level functions that are independent of the underlying network infrastructure. NGSON aims to bridge the service layer and transport network over IP infrastructure to address the accommodation of highly integrated services. NGSON particularly focuses on composing new collaborative services by either using existing components (from IMS, NGN, SDP, etc.) or defining new NGSON components. Key functional entities of NGSON service control include service discovery and negotiation, service routing, and service composition. Web service technologies have been employed to realize these functional entities [60], [61].

This analysis on network service exposition shows that recent evolution of service management in telecommunications has followed a path towards network virtualization through decoupling service provisioning from data transport and exposing network infrastructure through resource abstraction. The SOA principle and Web service technologies still play a key role in facilitating virtualization in Telco networks. Let us study next some projects on network virtualization and network slicing.

2.4.2 Network Virtualization

A precise definition of "Network Virtualization" is elusive. Ideally and in its theoretical definition, network virtualization presents the abstraction of a network that is decoupled from the underlying physical equipment where multiple virtual networks can run over a shared physical network infrastructure and each virtual network can have a much simpler (more abstract) topology than the underlying physical network.

Although network virtualization has gained prominence as a use case for SDN, the concept predates SDN and has in fact evolved in parallel with programmable networking. A complete history of network virtualization would require a separate chapter. Therefore, we survey some network virtualization projects where we focus on developments that relate directly to programmable networks. The two technologies are in fact tightly coupled. Programmable networks often presumed mechanisms for sharing the infrastructure across multiple tenants and supporting logical network topologies that differ from the physical network. Both are central to network virtualization.

Based on that, network virtualization proposes a set of architectural principles [62]:

Coexistence: Coexistence of multiple Virtual Networks (VNs) is the defining characteristic of a network virtual environment [63], [64], [65]. It refers to the fact that multiple VNs from different tenants can coexist together, spanning over part or full of the underlying physical networks provided by one or more network infrastructure providers as shown in Figure 2.8.

Recursion: When one or more VNs are spanned from another VN creating a VN hierarchy with parent-child relationships, it is known as recursion as well as nesting of VNs [66]. Service provider 1 (SP1) in Figure 2.8 leased away a portion of its allocated resources to Service provider 2 (SP2), to whom it appears simply as a virtual network infrastructure provider.

Inheritance: Child VNs in a virtual network environment can inherit architectural attributes from their parents, this means that the constraints on the parent VN automatically translate to similar constraints on its children [66].

Revisitation: Revisitation [67] allows a physical node to host multiple virtual nodes of a single VN. Use of multiple logical routers to handle diverse functionalities in a large complex network allows a service provider to logically rearrange its network structure and to simplify the management of a VN. Revisitation can also be useful for creating testbed networks.

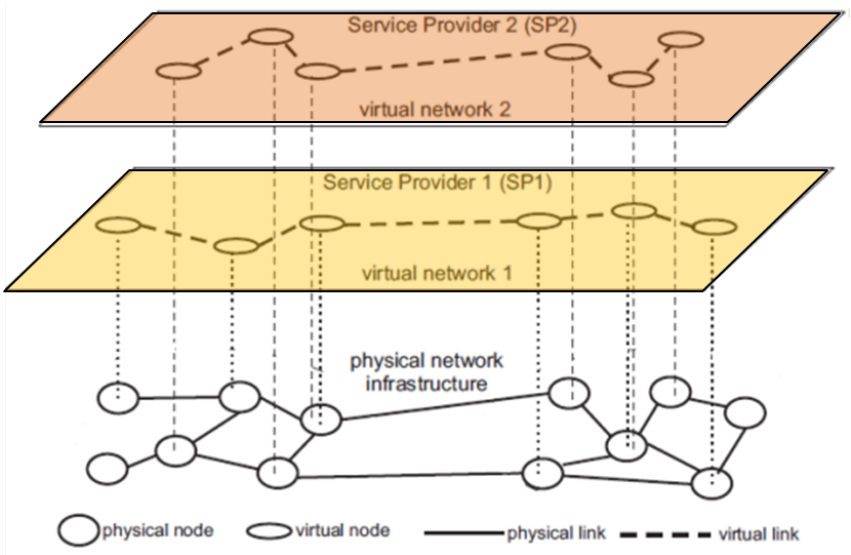

Figure 2.8 – Representation of a Network Virtualization environment

Multiple research projects aimed at realizing network virtualization environments that respect these principles. We list and describe few of them, then compare them according to three network virtualization characteristics.

4WARD project (Wired and Wireless World Wide Architecture and Design for Future Internet) [68]: 4WARD project was an attempt to implement an on-demand instantiation of virtual networks over heterogeneous network infrastructures (wired, wireless and optical networks). It defined three main business roles to simplify the network instantiation management:

1. Infrastructure Provider (IP): management of underlying network resource,
2. Virtual Network Provider (VNP): responsible for virtual networks' creation, and
3. Virtual Network Operator (VNO): in charge of configuring virtual network services

GENI (Global Environment for Network Innovations) [69]: GENI proposed test platforms to evaluate new network architectures generating real users' traffic flows. Virtual resources proposed in this project are presented as "network slices" and apply time and space sharing.

PlanetLab [70]: PlanetLab offered test platforms to design, evaluate and deploy service-oriented network architectures. This project also applied slicing by assigning virtual node slices to multiple applications and proposed distributed control policies to control each slice.

VINI [71]: Based on PlanetLab initiative, VINI project proposed virtual network infrastructure allowing researchers to evaluate and test new protocols and services. PL-VINI [71] et Trellis [72] are the main prototypes of this project.

UCLP (User Controlled Lightpaths) [73] [74]: UCLP aimed at defining a control and management system for virtual optical networks. It presents network resources as software objects. These objects can be used to create a personalized logical network. UCLP defined three management layers. User Access Layer to allow users and network administrators to access and configure UCLP resources; Service provisioning Layer to manage service provisioning; and Resource Management Layer to manage logical and physical resources.

CABO (Concurrent Architectures are Better than One) [65]: CABO separates network infrastructures' provider from network services' provider to create new network services independently from the underlying infrastructure. The project exploited virtualization techniques to allow network service providers to run simultaneously multiple end-to-end services over network resources from different network infrastructure providers.

We have analyzed these network virtualization projects based on three main characteristics:

Virtualization Layer: is the OSI model layer at which virtualization has operated.

Virtualization Granularity: is the network granularity level at which virtualization is applied. It can be either, at a node level, link level or applied at both.

Abstraction Level: the level of abstraction at which virtualization is applied. Four levels of abstractions can be made in a network: hardware, network, service or data level.

We have noticed that these research projects have approached network virtualization in a layered manner. As a result, many projects have attempted to virtualize different layers of the network stack, starting from the physical layer (L1) up to the application layer (L7). These projects also applied virtualization at two granularity levels, nodes or links and some of them were applied to both, nodes and links. However, the abstraction level is the hardware level for all of the approaches. We can conclude that the hardware abstraction level for virtualization is already correctly realized. Network virtualization approaches target the lowest abstraction level and virtualize the physical nodes and physical links.

We survey in the next Section Network Slicing, another form of network virtualization.

2.4.3 Network Slicing

In conventional networks, virtualizing a network functions may be a complicated task. Each virtual component needs to run its own instance of control plane software. In contrast, virtualizing a network network function in an SDN network may be much simpler. The control plane software is decoupled and the network function instance is independent. This is commonly known as SDN-based network virtualization or network Slicing. Slicing concept has been introduced in earlier work on PlanetLab [70]. A slice has a pool of shared network resources and is managed by a defined SDN controller.

In this direction, a number of research works have been conducted. FlowVisor system [75] proposes to enable a campus to support a testbed for networking research on top of the same physical equipment that carries the production traffic. The main idea is to divide traffic flow space into *Slices*. FlowVisor, as well as OpenVirteX [76] run as a hypervisors using OpenFlow between each SDN controller and the underlying network elements. The recent work in [77] has proposed slicing control of home networks, to allow different third-party service providers to deploy services on the network without having to install their own infrastructure.

A more recent work proposes ways to present each slice of an SDN network with its own logical topology and address domain [78] [79]. AutoSlice [80] is another SDN-based virtualization proposal. It focuses on the automation of the deployment and operation of virtual SDN (vSDN) topologies with minimal mediation by the substrate network operator. Additionally, AutoSlice targets also scalability aspects of network hypervisors by optimizing resource utilization and by mitigating the flow-table limitations through a precise monitoring of the flow traffic statistics. Similarly to AutoSlice, AutoVFlow [81] enables multi-domain network virtualization. However, instead of having a single third party to control the mapping of vSDN topologies, as is the case of AutoSlice, AutoVFlow uses a multiproxy architecture that allows network owners to implement flow space virtualization in an autonomous way by exchanging information among the different domains.

FlowN [82] [83] is based on a slightly different concept. FlowN is analogous to a container-based virtualization, i.e., a lightweight virtualization approach, while FlowVisor can be compared to a full virtualization technology. FlowN was primarily conceived to address multi-tenancy in the context of cloud platforms. It is designed to be scalable and allows a unique shared controller platform to be used for managing multiple domains in a cloud environment. Each tenant has full control over its virtual networks and is free to deploy any network abstraction and application on top of the controller platform. The common point for all these Slicing solutions is that they rely only on SDN-based virtualization of networking resources.

We provide in the next Section, a landscape of the architectural solutions regarding the evolution of networks through SDN and NFV integration.

2.4.4 Architectural Integration of SDN and NFV

It is known that standardization bodies define specifications and implementation guidelines for SDN and NFV seperatley. The Open Networking Foundation (ONF) [84] is the organization dedicated to the promotion and adoption of SDN through the development of the OpenFlow specifications [85] [86]. The ETSI NFV ISG [43] is the standardization initiative regarding NFV specifying general and scalable architectural and operation solutions to meet challenges placed by NFV. The defined architecture has introduced typical virtualization aspects, like the transformation of management and orchestration of VNFs. The common challenges to both physical and virtualized network functions, such as the control and operation of the end-to-end network service, were not considered.

Later, the convergence of both technologies became also a subject in standarization bodies and in different research works. The integration of the networking aspects of SDN into NFV architectural framework are studied.

Indeed, both ONF and ETSI NFV ISG are already working to propose an SDN-aware NFV [87] which offers the network the dynamicity to support new network-aware service provisioning paradigms and "A Flexible NFV Networking Solution" [88]. The solution proposes a combination of NFV and SDN for an operator implementation.It outlines the benefits of using an OpenFlow-enabled SDN approach for the NFV deployment in order to deal with the dynamic provisioning of networking services. In the IETF, standardization efforts around SDN and NFV include the Service Function Chaining (SFC) [89]. SFC Working Group is aiming at addressing the dynamic specification and instantiation of an ordered list of instances of service functions and steering flows through those service functions. Service Function Chaining mechanisms are considered in NFV to connect virtual network functions and thus the VMs containing them. Chaining mechanisms are based on the notion of Forwarding Graphs, equivalent to a set of Service Forwarding Paths (SFP) that describes the path that packets should follow to be processed through the chained VNFs. The work in [90] has proposed an analytic model for the NFV forwarding graph and aims at optimizing the execution time of the network services deployment. In IRTF, the Network Function Virtualization Research Group (NFVRG) [91] aiming at developing new architectures, systems, and software, and to explore trade-offs and possibilities for leveraging virtualized infrastructure to provide support for network functions. However, to the best of our knowledge, these standardization efforts are still poorly integrated.

Integrating NFV management with SDN control capabilities has also been the subject of research works giving high-level guidelines and architectural views.

The work presented in [92] resents evolution of NFV from the initial SDN-agnostic initiative to a fully SDN-enabled NFV solution, where SDN is not used only as infrastructure support but also influences how VNFs are designed. The work in [93] argues the need of conceiving a Network Operating System (NOS) to cope with the lack of network-wide abstractions, thus favoring the foundation for true network programmability. However, it does not handle in detail the integration of network control capabilities with NFV management functions.

The work in [94] present a three-layer architectural view for both network and cloud domains (infrastructure, control and application layers). The integration of these two architectures is discussed by introducing an orchestration functional box used for services that require a combination of these resources. However, its interaction with cloud and network control functions is only briefly described. T-NOVA European Project [95] [96] has analyzed benefits and challenges related to the adoption of NFV paradigm toward the 5G cellular framework. The work proposes a move from a high-level architecture to support the dynamic provision of VNFs on-demand and as-a-service. However, the authors provide only an introductory and conceptual description of the architecture.

The following works provide deeper insights into architectural design and related implementation issues.

The UNIFY project aimed at extending the potential of virtualization and automation across networking and cloud infrastructures by proposing a unified programmability framework [97]. It presents a functional architecture supporting automated, dynamic service creation leveraging NFV, SDN, and cloud virtualization techniques [98]. However, UNIFY does not explicitly refer to SOA principles, although some similar concepts exists, such as the description of services at different levels of abstractions (i.e., service graphs and network function forwarding graph). Nevertheless, UNIFY does not aim at supporting a multi provider marketplace of network services as this work proposes using SOA practices, which were, in fact, designed to ease the operation of multi-provider environments where services can be dynamically-advertised and discovered. The work in [99] proposes a modular VNF architecture providing policy-based management of VNF and service chains. Their main contribution consists in a basic ontology-based information model to describe network resources, network control functions, and VNFs capabilities with a uniform language. However, the role of network control functions (i.e., SDN controller) in the architecture and its interaction with the NFV orchestrator are not explained. The work in [100] addresses the problem of SDN-based virtual connectivity over multi-technological domains. Therefore they propose an integrated SDN/NFV management and orchestration architecture which is specifically conceived for the dynamic deployment of Virtual Tenant Networks and the related SDN Controllers (implemented as VNF in DCs). However, they do not address the service chaining problem. The work in [101] proposes the Cloud NFV Platform for enabling a telecom operator to deploy and manage service functions in a distributed cloud infrastructure. Although they focus on service chaining, the proposed architecture specifically reflects the software design of a prototype instead of providing general functional specifications. Some functional blocks are defined in terms of software implementation OpenStack and OpenDayLight. Moreover, they do not explicitly refer to service oriented principles and architectural guidelines.

Furthermore, the work in [102] stresses the importance of a description model for network service. Although they do not explicitly refer to SOA guidelines, this approach is in principle compliant with those guidelines. However, their contribution consists in a straw man model for service descriptions, not on architectural frameworks.

In the objective of promoting Telco-OTT interactions, SDN solutions have been proposed to address the challenges that operators face in today's business environment. In this respect, the ONF has been active and has defined an SDN-based solution: "Operator Network Monetization Through Openflow-Enabled SDN" [103] promoting SDN as a framework to enable symbiotic linkage between the operator's network and the applications that use the network to deliver end-customer services as shown in Figure 2.9. Several examples of network monetization were presented, including Bandwidth on Demand, Bandwidth Exchange, Pay for QoS, and Network Features for Pay. However, the enablers for openness in these solutions rely only on SDN. The above listed initiatives on the combination of SDN and NFV were not considered.

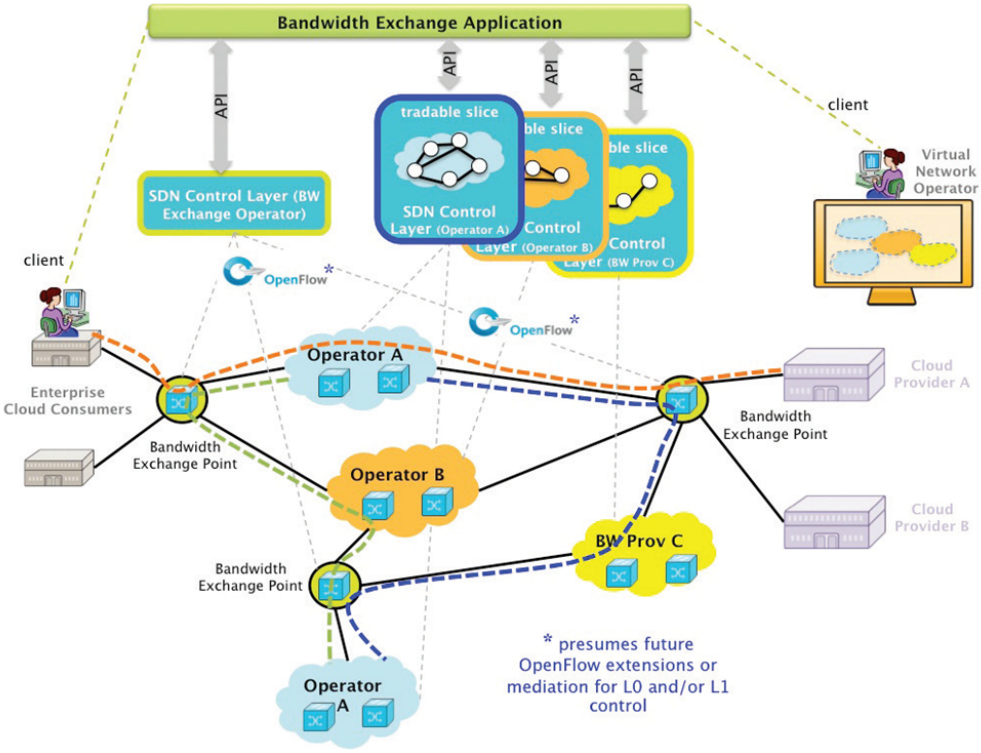

Figure 2.9 – Network Monetization through OpenFlow-Enabled SDN

This related work shows a clear momentum for exploiting SDN and NFV integration for the conception of more dynamic and flexible Telco network architectures. It also highlights the benefits of investigating the potential of such technologies to promote Telco-OTTs network interactions to achieve "Network Openness".

Based on these analysis, we highlight next, the challenges to be addressed in network exposition and network virtualization for an SDN-NFV Telco architecture enabling as-a-Service paradigms, network service exposition and orchestrtaion.

2.4.5 Summary and Challenges

Our state-of-the-art analysis confirms the need of reference architectural frameworks supporting the integration of network service orchestration and network control functions towards dynamic deployment and provisioning of network services in virtualized Telco environments. As SDN and NFV become important in order to build flexible networks, needs for automation (at data, control plane functions and management operations) and for dynamicity in the service deployment and life cycle management, become primary requirements. The main limitations are related to the level of abstraction adopted for the specification of the reference architectural framework. High-level conceptual guidelines and architectural design coupled with implementation choices are provided. One of the challenges is to consider a functional architecture which is independent from the implementation and whose specifications clearly identify the role of the components and shape the interactions between NFV management and orchestration with SDN control and forwarding.

In Telco networks, the exposition approaches are divergent and strongly depend on the networking technologies. Even though SOA is adopted in different initiatives, the tight coupling nature of network services with the network infrastructures limits the space for evolution towards an effective exposition of virtualized network architectures using SOA. However, NFV with its management and orchestration is a resilient enabler for such an evolution. The main challenge is to decouple the exposition from the composition using adequate APIs different from existing vertical APIs. Vertical APIs target a composed service, built from a pre-defined sequence and triggers a pre-defined procedure while horizontal APIs may be exposed by service orchestrators which would have a global view of elementary network functions and would perform an event-based composition with loose links and a life-cycle management.

Regarding network virtualization and Slicing, network hardware abstraction level for virtualization is already correctly realized. Virtualization approaches are today limited to the hardware level as we have just seen. These approaches do not allow to define virtual networks completely independent from network infrastructures. How can we get rid of today's lack of agility when realizing network virtualization? Relying only on hardware abstraction levels does not allow to create customized virtual networks. We thus need to think of novel approaches to get over these virtualization approaches that are stuck to hardware level and generalize it to include network functionalities. Therefore, another important challenge is to make network virtualization or Slicing rely on the virtualization of the network control plane. But how, can we rely on SDN and NFV to build dedicated control planes that would allow to apply and respect applications needs and their QoS constraints?

Furthermore, since the “as-a-Service" abstraction is considered a key element in the evolution of network architectures and networking technologies through SDN and NFV integration, another challenge is to take advantage of principles and practices that have emerged in the domain of service life-cycle management in the area of SOA design and implementation.

We propose to discuss our state-of-the-art analysis in the next Section to conclude this Chapter.

2.5 Conclusion

We have studied in this Chapter the research areas that define the scope of the thesis. We have analyzed the state-of-the-art and underlined the research issues that need to be tackled with respect to the thesis objectives.

First, in Section 2.2, we have considered component and service models in CBSE. Then, in Section 2.3, we have studied virtualization technologies with exposition and deployment approaches in Cloud Computing, the deployment in Telco Cloud through NFV and a focus on Cloud networking. Finally, in Section 2.4, we have studied network exposition, network virtualization and network Slicing, then targeted works on softwarized and virtualized architectures based on SDN and NFV integration. In this Section, we disscuss the convergence of the underlined challenges.

From our first analysis, we can conclude that designing network functions as services need to rely on a resilient models that define generic structures with functional and non-functional requirements. The functional requirements should allow to conceive new functional architectures with independent and composable services. The non-functional requirements should allow to accurately manage network functions and services. The design and implementation of network functions should also be separated as advocated by micro-services architectures in IT. The need for such a design goes with the Telcos openness need for exposing network functions like for applications in Cloud but within a Network Platform as-a-Service with a the specifically important challenge of considering QoS aspects in network functions exposition and management. For this openness objective, how these network functions can be exposed, knowing that this involves accurate description, discovery, selection and composition mechanisms?

In addition, the exposition of network functions need rely on the emerging SDN-NFV integrated architectures. The software design of network functions as services seems to be crucial for that. A network architecture that correctly integrates SDN control capabilities and NFV management capabilities would be a strong foundation for network exposition. Then, the description, discovery and composition of network functions designed according to SOA requirements should be also be integrated in such architectures.

Moreover, the deployment of network functions in an SDN-NFV architecture should be defined to be dynamic to allow an on-demand deployment in order to meet applications needs. The design of network functions as micro-services is strong enabler for that. With network virtualization in such architectures, full network abstraction would be also an enabler. There is a need to define the logical view of a virtual network to deploy. A logical network at the service level would allow to consider “network delivery" independently from the virtual network infrastructure level, i.e., the media delivery. The virtualization of the control plane would be a reliable research direction. Moreover, relying on the composition of network functions would make it possible to consider customization of network control functions in virtual network slices to have dedicated control planes.

Furthermore, orchestration of services should be considered along with as-a-Service modeling in order to define an automated life-cycle management of network functions, network services and virtual networks. This management should also integrated behavioral (QoS) management. But how to go forward from service composition to network service orchestration? what relationships should be made between the three interfaces of a service component (usage, control and management) with the three network planes (data, control and management) when re-defining the deployment, the invocation and life cycle management of network functions? How to manage service and QoS requests to ensure a compliance with the functional and non-functional services' requests operated by deployed network functions? In order to make network functions be dynamically re-configured in case of a change in the service request or in the QoS level, can the OSS/BSS be the only responsible for orchestration and management or do we need to define new levels?

In the next Chapters, we present our contributions for these main challenges.

3 Virtual Network Functions Modeling: VNF-as-a-Service

3.1 Introduction

NFV promises the ability to provide agile network services. It consists of implementing network functions, that are today performed by physical dedicated hardware, in the form of software function over virtualized infrastructures. However, as we have studied in Chapter 2, most of NFV-based implementations consist in cloudifying legacy network functions keeping the same service logic as defined by 3GPP specifications. Such virtualization clearly reproduce the management complexity for network services regarding deployment and provisioning.

Therefore, in order to leverage the benefits of NFV adoption, one of the challenges for Telcos is to reconsider legacy functional architectures and rethink the services' logic to make them less complex and less monolithic. For that, Telcos should adopt an optimal software designs for VNFs. The optimal software design is not immediately obvious but resilient enablers exist. software models are designed for services that need distribution over virtualized and Cloud infrastructures and VNFs are also meant to be distributed over virtualized Telco infrastructures. In addition, service components and micro-services bring agile management through composition, scalable elasticity, efficient use of resources, faster reconfiguration, a higher availability and cost-efficient redundancy. For the openness objective, modeling network functions as service components is crucial for the exposition in a Network-as-a-Service context.

We have considered this problem and we present, in this Chapter, our contributions for an "as-a-Service" software design for VNFs, then propose a "VNF-as-a-Service model" [BOU01] [BOU02] [BOU03]. We first define the most adequate service model through a design approach (Section 3.2). Then, we detail functional and non-functional requirements on the content and the structure of as-a-Service components (Section 3.3). Later, to complement the software deign, we propose an approach to rethink the legacy functional architecture of network services through functional decomposition (Section 3.4). In Section 3.5, we apply the as-a-Service model and the decomposition approach to IMS network functions to propose an IMS-as-a-Service architecture. Then in Section 3.6, we present the implementation and evaluation results of our proposition. Finally, we give our conclusions in Section 3.7.

3.2 Design approach for Components “as-a-Service"

We present in this Section an approach to design a software component as a service in order to make it compliant with SOA and micro-services service design requirements. We propose an approach that allows to cover progressively the required properties for its design. Our approach involves five steps.

3.2.1 *Step 1: To structure*

In a system where a service is available through the network and where the service is considered as a node of an architecture, we need to distinguish, and thus structure, the service. The structuring applies to services according to two aspects: the functional aspect representing the offered functionality and the non-functional aspect containing control functionality representing the automation and policies serving the management and control functionality. In Fractal approach, a component is represented as a “business" component with its “usage" interfaces as Figure 3.1 shows.

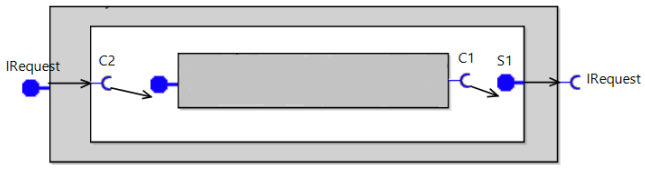

Figure 3.1 – Representation of a Component

A component needs to have control and management interfaces. That is why we propose to adopt the GCM structure. The resulting component structure at this step will be transformed to become a component including a management membrane with two interfaces: a control interface and a management interface. The Membrane is proposed in GCM and standardized by ETSI [6]. Thus, the service is represented with a functional part, i.e., the business content, and a non-functional part, i.e., the membrane.

3.2.2 *Step 2: To integrate*

To integrate a component into a global service environment, there is a need to link it to other components. Therefore, we propose he integration of a service through a definition of its functional and non-functional interfaces. It is related to the links that the interfaces provide. They allow the component to be invoked (functional) and managed (non-functional). Moreover, these interfaces allow to realize compositions based on the desired organization. This latter can be hierarchical, distributed or even centralized. Consequently, the service component is integrated in its service environment with: a business content with external interfaces representing the functional part, and a membrane for the non-functional aspects, with management and control interfaces to be linked to other components and to communicate within the service environment.

3.2.3 *Step 3: To self-control*

For the control aspect, we propose to embed a QoS Control agent in order to introduce the needed autonomic aspect in an environment that is meant to scale and is subject to become rapidly more complex. The aim of introducing autonomic control of components is to enforce monitoring mechanisms that collect information concerning the behavior of the component in order to control the respect of service level agreement (SLA) and QoS level and react in case of non-compliance. Thus, at this stage, the service becomes a self-controlled service. We rely on a recursive service architecture, where a service may comprise a set of self-controlled service components. Therefore, the service can be integrated within a global self-controlled service architecture.

3.2.4 *Step 4: To design as-a-service*

This step aims at ensuring that the offered service component can be added, removed or composed with other services, without crashing the whole organization, i.e., the global service architecture. The “as-a-service" design should allow flexibility in composing service offers and thus service customization, adaptability of services and solutions, as well as “on-the-fly" deployments. For this purpose, a set of properties needs to be satisfied for designing a “simple" service as an “as-a-service" service. The service design needs to rely on the following main properties: statelessness, loose coupling and reuse.

Statelessness: a service should perform the same processing to all requests without keeping any information about their data or their contexts. This allows a service to always offer the same function to all clients and requests. It should have a single type of interface and its the operations should be conceived to perform the processing without depending on information received during a precedent invocation.

Loose Coupling: the bindings or links between services in a service composition should be as loose as possible without any predefined sequence between services. This is to eliminate all types of functional coupling between services. Thus, loose coupling ensures a flexible composition of service components. Service composition consists of generating a global service by composing or chaining a set of elementary service components. The composition would thus be flexible for customization by adding, replacing, and removing service elements according to users’ needs.

In addition, for software engineering needs, the *reuse* property and is strongly recommended in this approach. *Reuse*: the possibility of reuse is needed to simplify the software development of services that meet the new needs. Services designed based on this approach and properties (as-a-Service) would be reusable thanks to the generic character of their interfaces (usage, control and management).

We detail other properties further in Section 3.3.

3.2.5 *Step 5: To describe*

A Service needs to be correctly described by the service provider and visible by users that would request it. For that, there is a need for service description and service registry using formal processes. These two properties allow establishing a service catalog. To design an application or a service, the architect selects multi-tenant services in the provider's catalog. Using the QoS-based as-a-Service model, the selection would be based on the specified offered QoS and thresholds values. At run-time, in order to introduce agility and eliminate static configurations, a service has to be invoked through standardized API. A service would include interfaces dedicated to QoS compliance control, service control and service programming.

3.2.6 Architecture of a component “as-a-Service"

Based on this approach, we define the architecture of an “as-a-Service" component as follows. Figure 3.2 illustrates the architecture of “as-a-Service" component. The architecture of an “as-a-Service" component inherits SCC component structure, interfaces and membrane. An “as-a-Service" component has two functional interfaces (in blue): a Server interface that includes the service methods (processing functions) that are performed by the service component, and a Client interface which invokes the following service in a composition transmitting the result of its processing to another component or for exploitation of the final result. The functions of these interfaces have no return value.

The triptych In Monitor, Out Monitor and QoS Control, associated to each service component, introduces an homogeneous autonomic management approach. Here, the idea is to position the Monitoring mechanism of the MAPE loop and integrate QoS monitors at the nearest level from components which are in fact services. This brings the most pertinent monitoring information about a component. The QoS measurements help notify in case of any QoS requirements violation. If any, a dynamic reaction to repair or replace the component is performed. The structure of the membrane allows components to have a generic characteristic and thus components' reuse. IConfigMnitor (In and Out), IConfigQoS and IActivate (in green in Figure 3.2) are non-functional server interfaces representing the management interfaces. They hold the necessary mechanisms to manage the configuration of non-functional components (the monitoring and QoS control components) in the membrane. The control interface QoSStatus is a client non-functional interface (also in green in Figure 3.2). It contains mechanisms to enforce the self-control information to the administrator in charge of reacting to QoS violation events. It outputs InContract notifications as long as the behavior is compliant to the SLA, otherwise it triggers OutContract. Absence of InContract events can be used to detect severe failures from the service component.

This step-by-step approach allows to design “as-a-Service" service components. We detail in the next section the properties of the “as-a-Service" model.

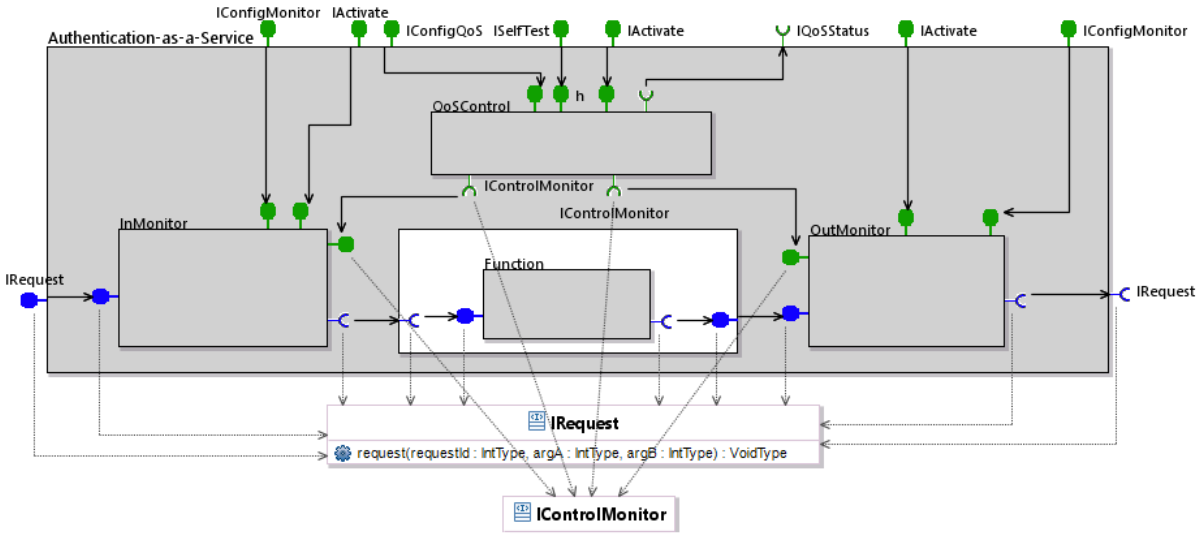

Figure 3.2 – Architecture of a component “as-a-Service"

3.3 Design requirements of “as-a-Service" modeling

The software design of services involves the software structure and the business service logic. “as-a-Service" components should satisfy a set of requirements in the form of properties.

We distinguish properties related to:

- The definition of the structure and the formal descriptions of service components, i.e. the nodes themselves,
- The definition or design of the service logic and functional architecture of service components, i.e. the interactions between service components,
- The management of the service components.

3.3.1 Properties related to service structure

Cohesion service components should be consistent. The service logic offered should be relevant and recognized as a meaningful business service for potential customers. The service rendered by the component should find all its functionalities in a logical way internally. We also call this feature: self-sufficiency, autonomy or even functional decoupling.

Reuse a service component should be reusable to build different services, in different compositions and different environments.

Abstraction Beyond service descriptions that should appear on service catalogs and SLAs, service components should abstract the internal service logic from outer service environments.

Invariance A service component should have an identical structure, that would not vary from a level to an other in a hierarchy of service components. That means that the structure is invariant when scalability and elasticity are needed. This property is expressed in components construction, derived from the Fractal core principles.

Statelessness A service is stateless if it processes each received request as an independent transaction without any relationship with previous ones. A service component should then neither keep information regarding its state or its processing state nor handle information about previous requests. If it maintains its state for a long period, it will lose the “loose coupling" feature, its availability for other incoming requests and even its possibility to scale. Each component should handle data coming only from outside its area of responsibility i.e., from other service components so that its functional behavior do not use data received from previous invocations. For that, we need to rely on transactions in unconnected mode that define well-specified formats of in-requests and out-responses. Here, the SCC structure with its interfaces would help. We also need to delegate information handling and state management to external entities. This feature is crucial as it impacts the independency of a service component and thus the possibility to include it in compositions that need to be dynamic.

Mutualization A service provider should offer the same service component instance as-a-Service to multiple users. By mutulization we mean multi-tenancy. Service components should support multi-tenancy in order to be invoked by multiple users requiring the offered service either simultaneously or not. This reinforces the statelessness and the loose coupling features. Mutualization requirement calls for a need for loose bindings or connections between service components to have the capacity to provision multiple users and answer multiple service requests autonomously. Thus, Mutualization will help realizing minimum functional coupling and loose coupling between functions.

3.3.2 Properties related to service interactions

Loose Coupling Service components should have no predefined sequence between them and should maintain relationship with minimized functional coupling.

Invocation A service component should be accessible and invoked based on service requirements in SLAs (invocation interface, function or service and QoS level). We distinguish three types of service contracts: syntactical contract (service interface, function, service or process name, input/output parameters and structural constraints), semantic contract (informal description of the function or service with service use rules and constraints), and service-level contract with (defines the service commitments, i.e., QoS and SLA parameters like time to access, to process, to response,...).

Composition Multiple service components should be able to be chained as elementary entities (primitive or composite components) to create a service. They should be effective service composition participants, regardless of the size and complexity of the composition. This composition requirement feature is verified only if all of the features described are verified.

3.3.3 Properties related to service management

Description Service components should be describable based on meta-data in an independent manner from their implementation specificity. The formal description should have a logical and meaningful structure.

Registration Service components should be able to be registered in a Domain Registry. This registration can be made through a publication (publish) of its service offering, QoS level and state. It should also be able to discover its environment through service discovery.

Exposition This feature includes cohesion, description, registration and invocation. Exposition is providing the functional and non-functional description of service components as well as their inherent QoS level offered through catalogs on service portals to allow a third-party actor to select and/or build a service based on his profile and competences .

Auto-management Service components should be able to monitor and control their behaviors (non-functional aspects) using autonomic management approaches like MAPE loop. Placing the monitoring of QoS very close around each service component and business logic helps to detect exactly the malfunctioning component.

Ubiquity ubiquity is the high equivalence between service components. Service components should be defined and described based on their core function and QoS level they offer (the values of QoS parameters). According to this definition, Service components may be grouped into communities of ubiquitous or identical service components where service components of a community provide the same service even if their business codes or algorithms are different, with the same QoS level. This feature goes with scalability issues, as the service provider may decide to scale the service by adding ubiquitous service components. It also enables higher service availability to find the requested service with the desired QoS level as this gathering in ubiquitous components community is an approach to set redundancy schemes.

These requirements apply to the software design of services on both, functional and non-functional aspects. Their generic nature allow a service architect to apply them on any service. Our objective is to design VNFs based on the as-a-Service design approach and these requirements. However, in order to design network functions as services according as-a-Service model, it is inevitable to first rethink legacy network functions, that are quite complex and monolithic, into independent elementary services through a functional decomposition. Therefore, we propose in the next Section a decomposition approach to move from legacy network functions into VNFs as service components.

3.4 VNF-as-a-Service Modeling From legacy network functions to VNF-as-a-Service

We propose to apply the proposed as-a-Service model for the software design of VNFs. But for that, we first need to reconsider the functional architectures and service logics of legacy network functions. Indeed, today's VNFs are the reproduction of legacy network functions over VMs, as we have seen in Chapter 2. The evolution towards scalable and cost-efficient resource consumption NFV requires a decomposition of today's network services into services representing sub-entities with finer granularity levels to eliminate their monolithic nature and the adoption of new functional architectures. The as-a-Service model is a resilient enabler for our objective of decomposing Telco network functions.

In this Section, we present our approach for the functional decomposition of legacy network functions. For an efficient decomposition of network functions, we describe a decomposition approach through three main steps.

Functional Decomposition

The first step is to identify all the functionalities that a network function has been designed to perform. It is important to list all the functionalities that an entity is responsible of in order to identify the functionallities that can be decoupled to extract them to create new ones apart. This allow to consider and accurately define the granularity level of functional entities by decomposing the logic into elementary **atomic services**. The new functional entities should perform one and only one service. However, it should be cohesive and consistent with a relevant service logic recognized as a meaningful business service for potential users (business models may determine granularity). It should be autonomous to find all its functionalities in a logical way only internally.

Separation of State

The seconf step in reconsidering the granularity levels of a network service or architecture is to consider separating the service logic of functional entities and the handled data or state. This is to build **stateless services**. A service is stateless if it processes each request as an independent transaction without any relationship with previous ones [104]. It should then neither keep information regarding its state or its processing state nor handle information about previous requests. Each service should handle data coming only from outside its area of responsibility. The service process and the data should be separated. Information handling and state management should be delegated to an external data base. With the existing approaches, it is impossible to terminate a VNF instance when it handles ongoing calls as this requires transferring the stored state to another VNF instance. Therefore, with stateless entities the logic of a service can scale easily by instantiating new stateless entities.

Functional Independence

The third step consists in defining the new service processes while verifying that services are **functionally independent and loosely coupled**. This means that services should have no predefined sequence between them. At run-time, they should maintain relationship with minimized functional decoupling. The loose coupling allows to verify the stateless and the atomicity requirements. Indeed, if a service maintains its state for a long period, it will lose the loose coupling requirement and thus its availability for other incoming requests and even its possibility to scale. Also if a service needs to rely on another service, then it is tightly coupled to it. For services composed of a set of elementary services, there should be reduced functional dependencies among interacting service components. These requirements are a must for building agile and scalable distributes services.

This approach provides guidelines for the functional decomposition and the design of VNF-as-a-Service. We rely on this approach to study the functional decomposition of IMS network functions as we present in the next Section.

3.5 IP Multimedia Subsystem (IMS)-as-a-Service Use-Case

Telcos have adopted IMS [54] as the de facto platform for service delivery in 3G and even 4G systems. We find it interesting to study its evolution for 5G towards a microservices-based architecture with a new software and functional design. In addition, most of NFV implementations have considered the virtualization of IMS through different approaches. This gives as comparison elements for evaluating our proposition comparing to recent research works. Therefore, we consider IMS as a case of study to illustrate the proposed approach.

In this Section, we first present in Section 3.5.1, a survey focused on research works that have targeted the virtualization of IMS and have considered the need for functional decomposition. Then, in Section 3.5.2, we present new functional entities for IMS-as-a-Service based on our approach for the functional decomposition of legacy network functions.

3.5.1 On IMS virtualization

Focusing on the IMS use-case, we observe that different research works have considered the virtualization of IMS. They mainly deal with the general problem of Cloud and Telco IMS integration [105].

Cloud Infrastructure for IMS

The research work presented in [106] proposes a cloud infrastructure for the IMS core network functions that run IMS entities on cloud-based VMs. It has proposed a resource allocation algorithm to scale IMS vertically and adapt to loads.

However, VNFs are exactly the legacy IMS functional entities (P-CSCF, S-CSCF, HSS,...) keeping the same service logic as defined today by 3GPP specifications. This means keeping also the stateful and highly coupled architecture of IMS functional entities. This does not allow horizontal scalability. It also proposes a resource allocation algorithm for high resource utilization. But, resource efficiency may not be maximal as functional decomposition into more granular entities has not been considered.

The work in [107] presents three IMS-as-a-Service architectural models for virtualized IMS based on NFV as follows:

Virtualized IMS

Virtualized-IMS architecture [107], like in [106], has implemented each legacy IMS entity as software on a single VM keeping the same interfaces between entities. It can somehow scale using the procedures standardized by 3GPP. However, the reproduction of the same functional service logic of IMS stateful architecture limits the scalability.

Split IMS

The Split-IMS [107] proposes a general architecture where IMS entities are split into multiple NFWorkers to be deployed on top of multiple VMs or containers. It also moves the state that these NFWorkers maintain to an external functional entity called Shared-Memory to make them stateless. However, even though it considers state separation and functional split, it does not propose a fine-grained decomposed architecture describing roles for NFWorkers. This is important to verify the optimality of the proposed split and how it affects the scalability, performance and resource efficiency.

Merge IMS

The Merge-IMS [107] deploys the four main IMS entities (P-CSCF, S- CSCF, I-CSCF, and HSS) into one IMS-VM. A new IMS-Locator entity assigns the subscribers to a dedicated IMS-VM at registration. HSS entities of each IMS-VM share a same database (SharedDB). Scaling is possible by instantiating a new IMS-VM with all entities. Thus, scaling is difficult due to the monolithic and stateful nature of IMS-VM.

These approaches perform re-engineering and do not address decomposition of functional entities. Also, they do not consider SOA or micro-services for the software design. This is an obstacle for agile management and elastic scalability. Indedd, the authors in [108] and [109] motivate the virtualization of IMS but critically review the architectures proposed so far for cloudifying the IMS and give research directions including reconsidering granularity levels of IMS network functions. Based on these needs to rethink network functions and service logic of network services, we propose next a functional architecture for IMS-as-a-Service.

3.5.2 IMS-as-a-Service Functional Architecture

In this Section, we present our proposition of IMS-as-a-Service architecture using the proposed VNF-as-a-Service design and decomposition approaches. It is the result of a functional decomposition of legacy IMS network functions [54] where we have reconsidered granularity levels of IMS network functions into new smaller, elementary sub-functional entities that can be reused, composed and mutualized, according to the proposed design requirements. Thus, this has involved separating the functional logic of entities from data and state.

In order to describe the proposed functional architecture, we formalize elementary functions and functional data. According to best practices in service architecture and software engineering, we suggest using stereotypical UML categories. We use the method defined in [110] to formalize the functional blocks as Service Functional Component (SFC) UML category, where **each SFC is a VNF-as-a-Service**, and the data as Service Functional Data (SFD).

Figure 3.3 presents a functional architecture for IMS-as-a-Service. It describes the new functional blocks and the data necessary for a registration network service of an end-user to the network. The network service of terminal registration consists in the establishment of a “temporary" link between an equipment connected to the network (under network coverage with an IP address) and a reachability address, such as a telephone number.

The proposed functional views describe examples of composed network services. The network service is defined via a composition of elementary functional VNFaaS micro-services. The execution of the service composition process is seamless. It does not take into account technical boundaries (the used terminal, platform, or network) neither responsibility boundaries. The functional views thus correspond to an end-to-end vision of a network service composed of VNFaaS micro-services.

Let us describe in details the services that are involved in the registration network service in our IMSaaS functional architecture.

Registration Network Service

The actors involved in the terminal registration network service are the end user, the network provider and the service provider. Please note that the network provider and service provider may be the same actor. The VNFaaS micro-services involved in the registration service are:

- *Launch terminal* SFC: it is used to initialize the terminal registration request. It uses the *network access parameters* SFD provided by the network. Typically this is an IP address as it is connected to the network.
- *Service attachment* SFC: receives the registration request and is responsible for establishing the attachment of the terminal to the service. This is a composite SFC, composed of *check registration session* and *authenticate account* SFCs which are elementary services.

  - The *check registration session* verifies if the user/terminal has an already registration session. For that, it checks the *registration session* SFD (such as a SIP Address Of Record) required for registration into HSS. If a session is already running and valid, then the *service attachment* retrieves *network access session* SFD and ends.
  - If there is no valid session, then there is a need for authentication. The *authenticate account* corresponds to the functional operation that checks access right to the network. It uses *credentials and account* SFDs (SIP login, password, IMPI, authentication keys,...) to authenticate the user. For that, it interrogates the HSS as it holds the initial account subscription. We detail the description of the authentication service further in this Section.

- *Terminal attachment* SFC: is responsible for the attachment of the terminal to end the registration service. It uses some SFDs and test results from the previous SFCs to finally register the terminal.

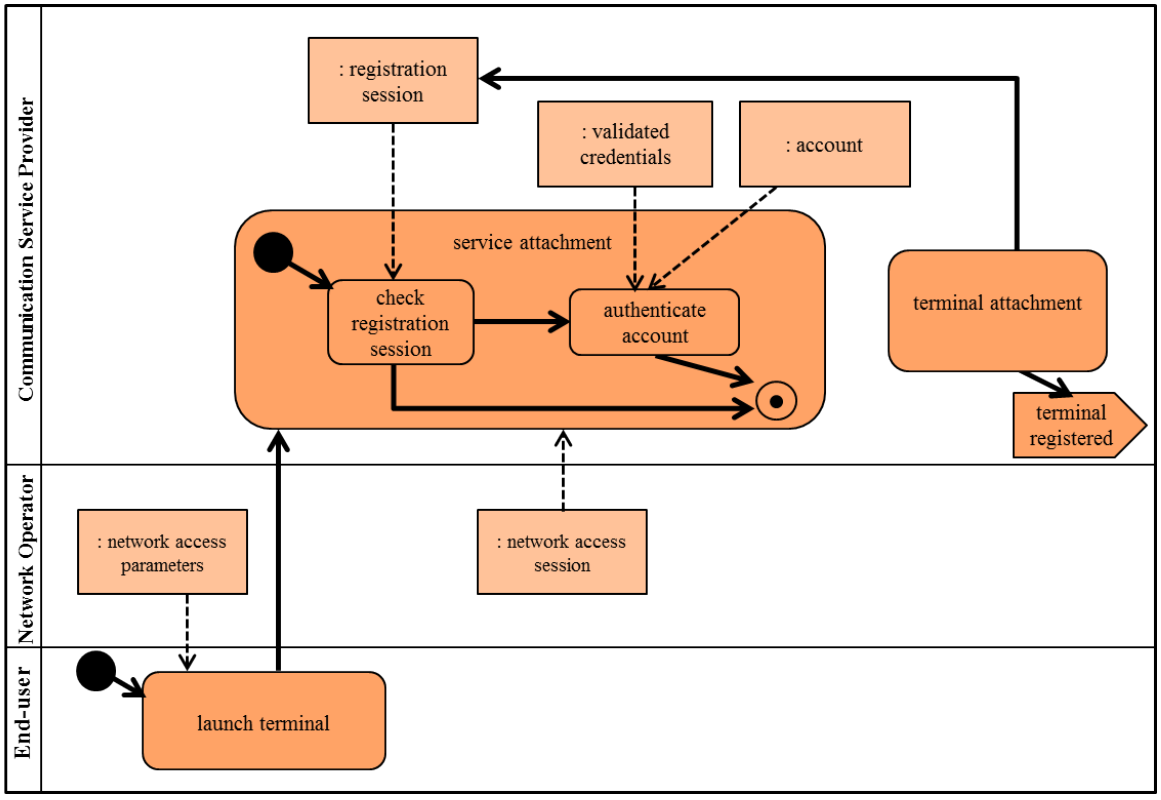

Figure 3.3 – IMS-as-a-Service: VNFaaS Micro-Services for Registration Service

We have chosen to focus on authentication and authorization services: on *authenticate account* composite SFC as it involves other elementary SFCs), and on authorization service through a scenario of making a call as we describe here after.

Authentication Service

Authentication relies in verifying the user identification, if the user is known by the network provider and service provider and if he has the right to access the network or service Figure 3.4presents the functional building blocks of *authenticate account* SFC. The SFCs involved in the authentication service are:

- *Check account* SFC: checks if the user is known and has a subscription (a user account and profile). For that it checks the *account* SFD within the HSS.
- *Select authentication means* SFC: is responsible for gathering SFDs required for the authentication and that are provided by all involved actors. That is: the user-provided credentials (login and password), the network access session and the Customer Premises Equipment (CPE) credentials as well as account SFDs.
- *Check credentials* SFC: is responsible for the verification of all the provided credentials by interrogating HSS.

After authentication, the *authenticate account* VNFaaS micro-service provides registration session SFD.

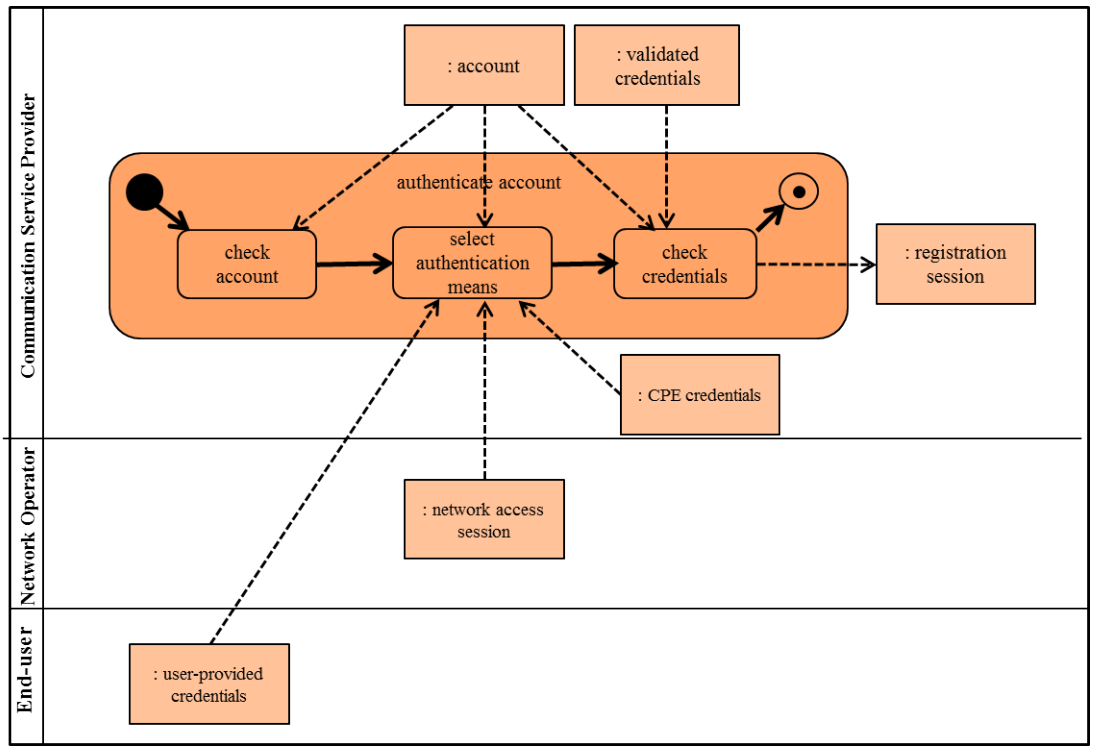

Figure 3.4 – Authentication Service in IMS-as-a-Service

Authorization Service

Figure 3.5 describes the process of a call network service. It involves authentication and authorization. At subscription, the service provider determines the profile and rights attached to a user's account stored in a data-base (HSS). Authorization service is performed when a user requests to access a service. It is performed only if the user has been authenticated by the service provider via a user account and credentials or through its terminal (e.g. a SIM card). Authorization is performed by *authorize call* SFC (see Figure 3.5). It is responsible of checking whether the user is requesting access to a subscribed service. Thus, it checks and controls the user rights to access a target service or a content according to its subscription profile by interrogating HSS to retrieve *requester account* SFD.

The actors involved are the caller, the callee and the service provider. When a consumer of the service wishes to communicate vocally and instantaneously with a contact, the caller user selects the telephone number and then makes the call, provided that he/she has the right to access the service. The service provider delivers the call, after having verified the validity of the callee's telephone number. If the callee answers, this produces a communication session that allows the caller and the callee to communicate. The verification of rights (authorization) is carried out at the time the call is requested.

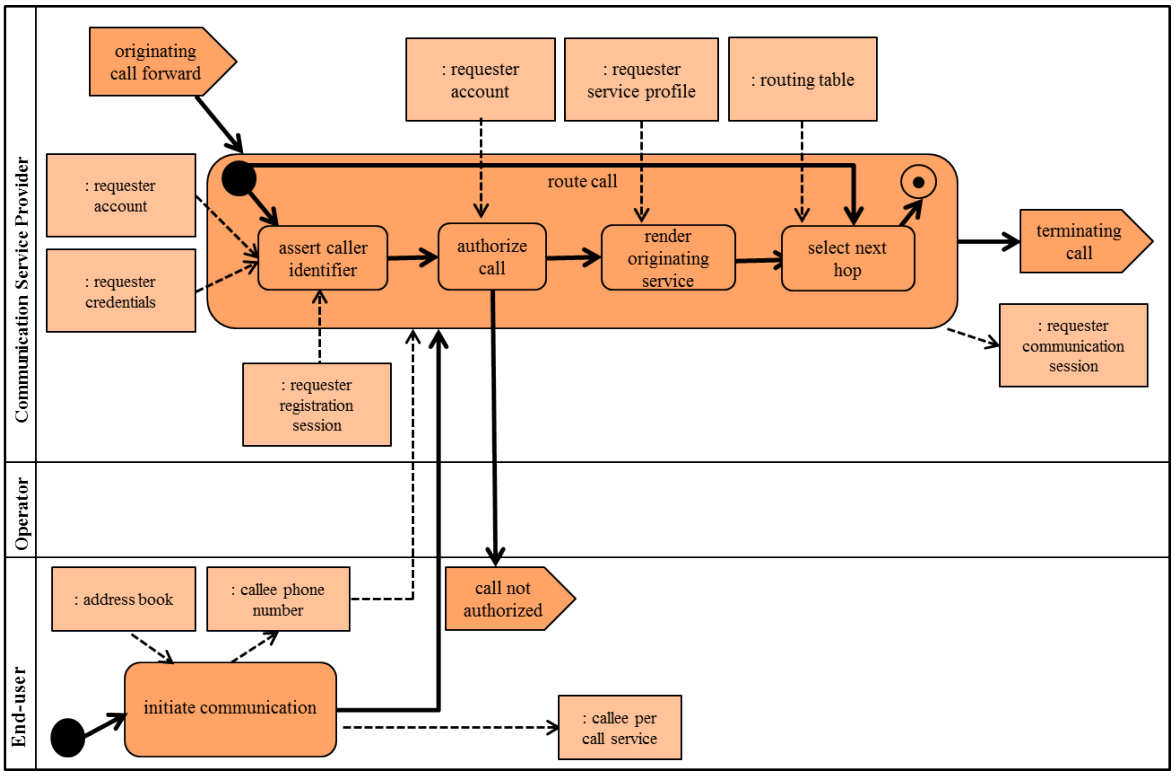

Figure 3.5 – Call Authorization Service in IMS-as-a-Service

We have implemented the proposed IMS functional entities according to VNF-as-a-Service design model as we detail in the next Section.

3.6 Implementation and Results

We present in this Section details about the implementation of the proposed IMS-as-a-Service architecture (Section 3.6.1). Later, we describe the evaluation metrics and the conducted experiments (Section 3.6.2). Finally, we give and discuss our evaluation results (Section 3.6.3).

3.6.1 Implementation

We describe in this Section the implementation environments and the tools we have used to develop the proposed services and architecture. In order to develop the proposed IMS-as-a-Service SFCs, we have first analyzed legacy IMS network functions codes from Fraunhofer FOKUS OpenIMS Core open source solution [111]. We have then developed the new VNF-as-a-Service functional entities proposed in the IMS-as-a-Service functional architecture of Figures 3.3, 3.4 and 3.5. We have developped IMS-as-a-Service functional entities in Java. We have used VerCors [112] platform for the software design to build the as-a-Service structure (envelope). VerCors is adapted for the modeling, specification, verification and validation of the software architecture of services. Figure 3.6 shows VerCors Eclipse development environment.

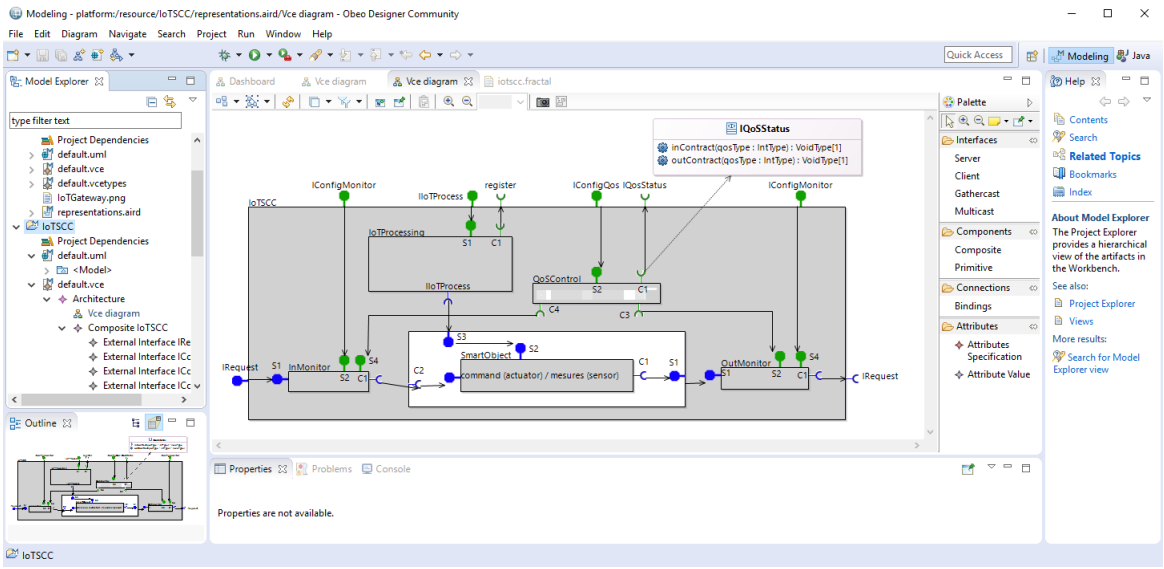

Figure 3.6 – VerCors Design Platform

Figure 3.7 below shows VerCors principles. VerCors allows a design through four phases:

- Diagram design on VerCors Component Editor (VCE) with classes and interfaces,
- Checking of the validity of the diagram,
- Generating Architecture Description (ADL) file and code template classes and interfaces,
- Adding the VNFs business code to the resulting code to have a final code ready for instantiation and execution.

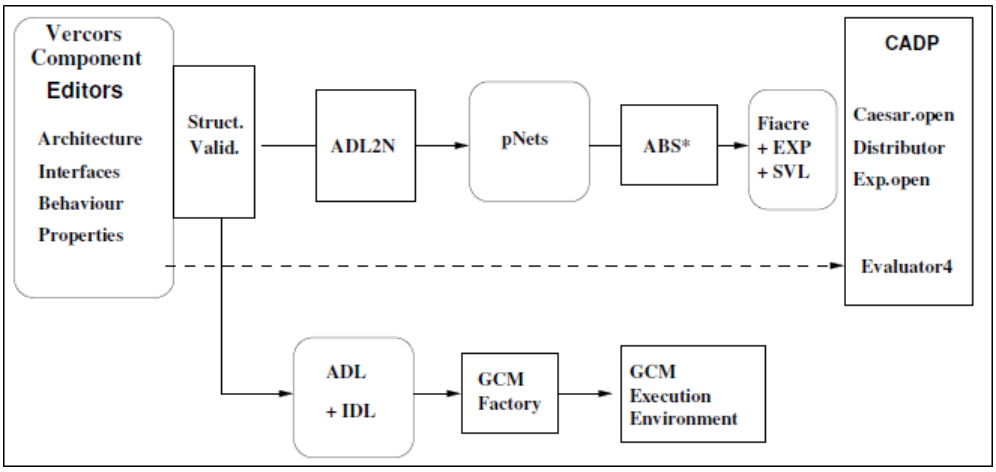

Figure 3.7 – Principles of VerCors Design Platform

After validation, we instantiate and execute IMS VNFaaS components using Proactive platfrom [113]. Proactive is an execution platform responsible of execution and life-cycle management of service components (start, stop, add, remove, configure,...). In our case, it plays the role of a VNF Manager and thus ensures the life-cycle management of IMS VNFaaS components. Proactive ensures the distribution of services over VMs of an experimental Cloud infrastructure called *Grid5000* [114]. This Cloud infrastructures implements OpenStack for the management of virtualized computing resources acting as a VIM.

Then, as the developed services are designed as stateless services, we have developed an external entity to handle state. This entity is called *Frontal*. It is a protocol automate and state automate. Indeed, in IMS, the SIP protocol [115] is a low level signaling protocol that needs expert knowledge. Current developments of IMS or VoIP services are usually based on the use of a SIP stack which implements the behavior of a SIP UA. Therefore, to have a stateless service environment, the interaction between VNFs should no longer use IMS 3GPP interfaces and protocols. We needed to hide and even eliminate the SIP complexity. The overall objective behind *Frontal* is to keep state outside VNFs and thus make them protocol-independent.

We have installed a VoIP call generator, StarTrinity SIP Tester [116], that acts as end users on a separate machine. It allows to simulate thousands of simultaneous VoIP calls. It sends SIP requests towards the *Frontal* external stateful entity. *Frontal* is responsible of converting SIP signaling messages (Register, Invite,...) into Java RMIs (Remote Method Invocation), lightweight reliable and scalable interfaces. It is also responsible of keeping state as an intermediate between end-user terminal and the IMS VNFaaS components. We have also set an experimental MySQL Data Base equivalent to HSS to store all the data (SFDs) necessary for our functional architecture. In addition, for evaluation objectives, we have set legacy IMS VNFs using Fraunhofer FOKUS OpenIMSCore open source solution.

We describe next, the experiments and measurements we have performed to evaluate the proposed solution.

3.6.2 Experiments and Measurements

Experimental scenario

In order to evaluate the proposed IMS-as-a-Service VNFs, we have performed the call service scenario of Figure 3.5 which involves the authentication service of Figure 3.4. We have simulated thousands of simultaneous SIP calls using StarTrinity VoIP call generator. SIP requests are sent towards the *Frontal* entity. *Frontal* which converts SIP messages into Java RMI and relays them to the first VNFaaS components of the service. During the execution of the service, SFDs are retrieved from the experimental MySQL DB.

Evaluation Method

The VNF service components we have developed are designed according to as-a-Service software design requirements. Therefore, they integrate the non-functional aspects we have proposed. Indeed, they integrate the self-control mechanism for QoS autonomic management detailed in Section 3.2.6. It is responsible for the monitoring of the behavioral (QoS) aspects of each service. A QoSControl component is associated with each functional component. This mechanism is also reproduced over a service composition in order to ensure an end-to-end QoS monitoring and management. The mechanism rely on four criteria to describe the QoS: availability, reliability, processing time and capacity. To evaluate QoS criteria, the InMonitor and OutMonitor retrieve a set of measurements, like the number of requests that a component receives, the time at which requests are received, the number of processed requests or the number of responses, the time at which responses are sent, etc. These parameters are then given processed by the QoSControl component to evaluate QoS criteria. We use this mechanism to measure the parameters related to QoS evaluation.

Evaluation metrics

During simulations, to evaluate the performance of our solution, we measure:

- The processing capacity through response time (or processing time) of authentication and authorization services for an increasing number of a pool of end users simultaneous call requests per second. The number of requests processed represents the number of users served simultaneously. The two services go through registration service. So, the total response time measured includes registration time.
- The resource consumption efficiency over time. We measure the consumption of memory and CPU by authentication and authorization services. The measurements here include the consumption of all VNFs (or micro-services) involved in each service, even though the ones involved in registration as they are also running.

We present next the evaluation results using these metrics and discuss them.