D3.1: NECOS System Architecture and Platform Specification. V1

Deliverable

Document ID	NECOS-D3.1
Status	Final
Version	2.3
Editors(s)	Stuart Clayman (UCL)
Due	31/07/2018
Submitted	08/10/2018
Updated	14/03/2018

Abstract

This deliverable presents an initial version of the NECOS System Architecture and the Platform Specification. After a discussion on slicing concepts in general, and reviewing the State of the Art in standardization and research projects, it provides an overview of the NECOS architecture, and a more detailed functional description. The components of the architecture are mapped to fundamental roles that an actor can take when participating in the NECOS ecosystem. Then each of the main architectural blocks which are in the three main high-level sub-systems are presented. This is followed by a section which presents the interactions between the functional blocks within the NECOS architecture.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CONTRIBUTORS

Contributor	Institution
Stuart Clayman	University College London (UCL)
Francesco Tusa	University College London (UCL)
Alex Galis	University College London (UCL)
Christian Esteve Rothenberg	University of Campinas (UNICAMP)
David Fernandes Cruz Moura	University of Campinas (UNICAMP)
Marcos Rogério Salvador	University of Campinas (UNICAMP)
Nazrul Islam	University of Campinas (UNICAMP)
Suneet Kumar Singh	University of Campinas (UNICAMP)
Antonio Jorge Gomes Abelém	Federal University of Pará (UFPA)
Billy Anderson Pinheiro	Federal University of Pará (UFPA)
Joan Serrat	Universitat Politècnica de Catalunya (UPC)
Javier Baliosian	Universitat Politècnica de Catalunya (UPC)
Ilias Sakellariou	University of Macedonia (UOM)
Panagiotis Papadimitriou	University of Macedonia (UOM)
Lefteris Mamatas	University of Macedonia (UOM)
Antonis Tsioukas	University of Macedonia (UOM)
Luis M. Contreras	Telefónica Investigación y Desarrollo (TID)
Augusto Neto	Federal University of Rio Grande do Norte (UFRN)
Marcilio Lemos	Federal University of Rio Grande do Norte (UFRN)
Silvio Sampaio	Federal University of Rio Grande do Norte (UFRN)
Marcelo Ribeiro Nascimento	CPqD Telecom Research and Development Center
Douglas Salles Viroel	CPqD Telecom Research and Development Center
Alisson Medeiros	Federal University of Rio Grande do Norte (UFRN)
Sand Luz Correa	Federal University of Goias (UFG)
Leandro Alexandre Freitas	Federal University of Goias (UFG)
Paulo Ditarso Maciel Jr.	Federal University of São Carlos (UFSCar)
Fábio Luciano Verdi	Federal University of São Carlos (UFSCar)
André Beltrami	Federal University of São Carlos (UFSCar)
Rafael Pasquini	Federal University of Uberlândia (UFU)

Reviewers

Reviewer	Institution
Augusto Neto	Federal University of Rio Grande do Norte (UFRN)
Javier Baliosian	Universitat Politècnica de Catalunya (UPC)
Alex Galis	University College London (UCL)
Raquel Lafetá	Federal University of Uberlandia (UFU)
Francesco Tusa	University College London (UCL)

Acronyms

DC	Data Center
VIM	Virtual Infrastructure Manager
WIM	Wide-area network Infrastructure Manager
VM	Virtual Machine
VNF	Virtual Network Function
WAN	Wide-Area Network
VNFM	Virtual Network Function Manager
LSDC	Lightweight Software Defined Cloud
KPI	Key Performance Indicator
MEC	Multi-Access Edge Computing
NFV	Network Functions Virtualization
SDN	Software Defined Networking
OTT	Over The Top
URLLC	Ultra Reliable and Low Latency Communication
mMTC	massive Machine Type Communication
eMBB	enhanced Mobile BroadBand
SLA	Service Level Agreement
SLO	Service Level Objective
CDN	Content Distribution Network
IMA	Infrastructure and Monitoring Abstraction
SRO	Slice Resource Orchestrator

Executive Summary

The NECOS architecture is designed to provide slices as a service. Each of these slices is a set of computational and networking resources created with the purpose that customer users (tenants) can run one or several application services on top. To ensure that the supported services can be deployed appropriately, these slices will be orchestrated to fulfill on-demand end-to-end quality requirements, independently each one from the others, and spanning across a shared infrastructure of several administrative domains. The slicing in NECOS can be performed either at the resource level or the virtual infrastructure management level of each of the administrative domains contributing to a slice. The high-level NECOS architecture can be described in terms of its constituting functions, and from that point of view, four functional domains can be distinguished, namely the Tenant's Domain, the Resource Domain, the Resource Marketplace, and the Slice Provider.

The Tenant's Domain contains the functionality needed by the slice users to request the creation/teardown of slices and to order the deployment of services. The Tenant's Domain interacts with the Slice Provider through an API named Client-to-Cloud Interface. The way by means of which the interaction between these two functional domains is produced depends on the tenant's preferences, and it can range from the detailed specification of the required slice resources to the specification of the service attributes, leaving, in that case, the Slice Provider the responsibility to create the appropriate slice.

The Resource Domains contain the physical and virtual resources that are parts of a given slice. Each Resource Domain may rely on infrastructures of edge data centers, central data centers, wide area networks or any combination of the above. Whatever the type of resources it may have, an entity wishing to participate in NECOS must: (i) exhibit its own capabilities for the on-demand creation of virtual resources with a given capacity as well as the virtual managers associated to them; and (ii) the capability to offer the control of those virtual resources to the Marketplace and their management to the Slice Provider by means of defined APIs. In particular, the Resources Domain interacts with the Resource Marketplace through the Slice Marketplace Interface. In addition, the Resources Domain interacts with the Slice Provider through four types of interfaces, namely the Slice Instantiation Interface, the Slice Runtime Interface, the VIM/WIM Control Interface and the VIM/WIM Monitoring Interface.

The Resource Marketplace is the entity that will facilitate the Resource Domains to offer their resources to participate in slices and the Slice Providers to discover such resource offers. Three types of marketplaces have been foreseen in NECOS. With no particular order of preference, the first one is between telco providers, the second is within pre-established trusted parties, and the third one is a public one, open to everybody. A given Resources Domain can participate in any of the above marketplace types with an appropriate resource filtered offer. The Slice Broker stands as the key element in the Resource Marketplace. The Brokerage can be done either in push-mode, where the Slice Broker contact the Resource Domains to get specific resources or in pull-mode, where the Slice Broker searches for resources in a pre-established resource catalogue. The Resource Marketplace interacts with the Resource Domains through the Slice Marketplace Interface. In addition, the Resource Marketplace interacts with the Slice Provider through the Slice Request Interface.

The Slice Provider is the core of the architecture as it constitutes the Lightweight Software Defined Cloud (LSDC) platform. The functionality of the LSDC is meant to receive tenant's requests for the creation of end-to-end slices, take care of the whole slice lifecycle (creation, runtime monitoring and control, and decommission) and facilitate the deployment of the service components (i.e., virtual machines) on top of the slice infrastructure. The LSDC may receive slice creation requests in different granularity and formats. For that reason, it has to process such slice requests to make a common slice specification. Once this has been achieved, the LSDC has to interact with the Resource Marketplace Domain to get Slice Parts from different Resource Domains. These Slice Parts have to be then assembled to create the end-to-end slice. A model of the particular topology of each slice is kept in a database for the whole lifetime of the slice. The LSDC will have to track the behaviour of the slice in terms of a set

of KPIs through a monitoring module and trigger a vertical/horizontal escalation when needed. Once the slice is already operational, the LSDC will be able to receive the request from the Tenant's Domain to deploy the service components. These requests can also be of different formats, and for that reason, the LSDC will have to include appropriate adaptors. The coordination of all the above described functions will be conducted in the framework of the slice orchestration function.

1 Introduction

Currently, evolutionary alternatives for network operators and service providers such as the Telco Cloud and the Multi-Access Edge Computing (MEC) have emerged to extend the capillarity of the existing centralized data centers, enabling new environments of geographically spread cloud capabilities. The availability of computation, storage, and network resources as well as the kind of workloads and services to be supported differs from the traditional ones leveraging on the large centralized Data Centers (DCs). This fact imposes the need for new mechanisms to manage and control the computing infrastructure for this new kind of demands.

The concept of multi-tenancy is key to the evolution of the networks and cloud environments, enabling novel network slicing ideas on top of the existing telecom networks. The paradigms of network virtualization, mainly based on the Network Functions Virtualization (NFV) [RFC8172] approach, and network programmability through Software Defined Networking (SDN) [RFC7149], have tremendously fostered this evolutionary view, appearing as tools leveraging the implementation of slicing. However, these two paradigms are not sufficient for network slicing. Especially for the former, there is a prevalent NFV-centric view in the state-of-the-art related research projects and in the industry and which does not leverage a native integration of cloud computing and advanced networking. Overloading the NFV orchestration and management artefacts with slicing mechanisms without taking into account the cloud and the network perspective could lead to inefficient partitioning. This problem tends to become further exacerbated as new telecom services (5G services) are expected to be provisioned over multiple-technologies and spanning across multiple operators.

To help in fulfilling the requirement of making such an innovation for 5G services more reliable, faster, and simpler, slicing is associated with the partitioning of resources, and being able to create and redefine these partitions as needed. A slice is a grouping of physical or virtual (network, compute, storage) resources which can act as a seemingly independent sub-cloud, sub-network and can accommodate service components.

The NECOS project aims to address this innovation in a lightweight manner by means of a service and infrastructure management platform, called the Lightweight Software Defined Cloud (LSDC) platform. The LSDC platform exhibits the following three properties: (i) provisioning of service-agnostic slices but adapting the slices to the desired service characteristics; (ii) automating the process of slice configuration in multi-cloud environments; and (iii) providing a uniform management with a high-level of autonomicity.

The first property (i), will be achieved by developing adaptive orchestrators in charge of ensuring that the attributes prescribed to the network are also propagated into the (possibly heterogeneous) data centers that host the services. Tools such as Cloudify [IN1], Terraform [IN2] and Scalr [IN3] are currently used to orchestrate multi-cloud environments, but they lack essential features such as the creation of separate cloud slices to be used in isolation by different customers (tenants) to provide effective service elasticity. Similarly, research projects such as 5G Exchange (5GEx) [IN4] and UNIFY [IN5] offer capabilities to orchestrate network services over multiple administrative domains (multi-operator) but they are too NFV-centric.

NECOS will achieve the second property (ii), by designing a mechanism that partitions the underlying distributed infrastructure into constituent slice parts, and then combines these parts into a full slice. Finally, the third property (iii), will be achieved by extending existing management functionalities to: (i) discover suitable slice parts from a set of participating resource domains; (ii) deploy virtual elements over the full slice; and (iii) monitor the resources and services inside the slice.

The NECOS LSDC has the following important factors:

1. It empowers a new service model – the Slice as a Service, by dynamically mapping service components to a slice. The enhanced management for the infrastructure creates slices on demand and slice management takes over the control of all the service components, virtualized network functions and system programmability functions assigned to the slice, and (re)configure them as appropriate to provide the end-to-end service.

2. It enables easy reconfiguration and adaptation of logical resources in a cloud networking infrastructure, to better accommodate the QoS demand of the Slice, through using software that can describe and manage various aspects that comprise the cloud environment.
3. It allows each aspect of the cloud environment – from the networking between virtual machines to the SLAs of the hosted applications – to be managed via software. This reduces the complexity related to configuring and operating the infrastructure, which in turn eases the management of the cloud infrastructure.
4. The LSDC platform will offer the ability to a specific cloud provider to federate his own infrastructure with other cloud providers with different configurations in order to realize virtualized services through the use of the Slice as a Service concept. The users of the LSDC APIs and platform will be able to create virtual services that can span the merged cloud infrastructure offered by different cloud providers. This concept is not purely technical, it can also encompass business, cultural, geographical or in any other domain.

NECOS is a research project. Being a research activity means that there is not a pre-established path to follow to reach the final target. New paradigms or algorithms have to be pursued. Uncertainty is then consubstantial. The methodology relies on the Scientific Method, which starts identifying an observation domain where a given problem is identified, and the subsequent hypotheses are established to solve it. Those hypotheses are then validated or reformulated again through an appropriate campaign of experiments. At the same time, NECOS is an engineering project, more specifically a software engineering project. Therefore, it makes sense to establish a set of requirements that will have to be analysed, specified and validated.

To better understand the type of requirements to be considered, these will be coined in the context of a set of specific service scenarios. We undertake an analysis process to confer the LSDC with the main requirements based on its service agnostic properties. Immediately after the requirements elicitation, we start the design phase that entails the system architecture, its components, and interfaces. The system architecture is a critical step in the project lifecycle. Flexibility is a must in order to avoid decisions that can jeopardize the project outcome. For that reason, the system architecture will be conceived in two cycles. The first one is the overall architecture that leaves some aspects open to being fixed in the second one, which is taking into account the feedback from the implementation and the initial validation. These two last phases of the engineering process have to be carefully selected to provide enough information within the limitations of the available resources.

1.1 Structure of this document

This document describes the first cycle overall architecture of the NECOS platform, its main components, and their functionalities and is structured as follows. After this introduction, Section 2 summarizes the most important terms and concepts and an overview of the different levels at which cloud-networking slicing could be performed. Section 3 presents the NECOS architecture at its highest level of abstraction. It is constituted by the LSDC, the tenant of NECOS, the marketplace and the resource providers. The functionality of each one of these stakeholders and their mutual high-level interactions are presented. In addition, this section explains how the requirements elicited from several scenarios have been considered. Section 4 describes with more granularity the functionality and interfaces between the functional blocks characterizing the NECOS architecture. Section 5 is devoted to present the concept of roles in NECOS and how the functionality of these roles is mapped to the components of the architecture. In particular, this shows how a given entity can participate in the NECOS ecosystem. Section 6 describes the slice lifecycle in terms of the interaction of the functional components of the architecture and focuses on the most common of its workflows. The main part of the document concludes with Section 7. Finally, two additional appendices are provided. In Appendix 1 we discuss how a Data Center can be sliced and in Appendix 2 we make use of a feature-oriented requirements modelling technique to show how the requirements elicited in Deliverable D2.1 can be supported by the NECOS architecture.

1.2 Contribution of this deliverable to the project and relation with other deliverables

Deliverable D3.1 is a fundamental piece in the NECOS project because it documents the functional system architecture that unleashes the development of the project itself. In fact, the APIs within the scope of WP4 have to be designed according to what is stated in D3.1 in respect to the external interfaces of the LSDC platform. In addition, the proof of concept (PoC) prototypes to be conceived within WP6, showing the fulfilment of the LSDC characteristics, have to be based on components, to be designed within WP5, which materialize the functionality of specific building blocks of the NECOS functional architecture here presented. In summary, this deliverable constitutes the input of the key tasks of WP4 and WP5.

This deliverable can also be seen from the perspective of its input/output relationships with other deliverables of the project. The primary input of D3.1 is the Deliverable D2.1, where the different requirements to be fulfilled by the LSDC were formulated. We have to assure that the proposed architecture can facilitate all those requirements. The two main outputs of D3.1 are the APIs design within the scope of D4.1, and the functional components design within the scope of D5.1. In that respect, it is worthy to say that the components to be designed in D5.1 will not likely be the full-fledged modules of the architecture but only with the minimum functionality required by the PoCs conceived in D6.1.

The design of the NECOS architecture is an evolving process, which will take into account the lessons learned after the initial validation tests. For that reason, the Deliverable D3.1 cannot be considered as a final architecture proposal. Instead, it will be extended into the Deliverable D3.2 as soon as the inputs from the tests to be reported in the D6.1 are available.

2 Slicing concepts

Slicing is a move towards segmentation of resources and deployment of virtual elements for the purpose of enhanced services and applications on a shared infrastructure. Many groups including ITU, ETSI, IETF, and more are currently considering it.

Slices are expected to considerably transform the networking perspective and enhance software-defined architectures (e.g., SDN, NFV, etc.) by:

- Abstracting away the lower level elements, in various ways.
- Isolating connectivity at a sub-network level.
- Separating logical network behaviours from the underlying physical network resources.
- Allowing dynamic management of network resources by managing resource-relevant slice configuration.
- Simplifying and reducing the expenditure of operations.
- Support for rapid service provisioning.
- Support for NFV deployment.

Key characteristics of the Cloud Network Slicing include:

- The *concurrent deployment* of multiple logical, self-contained and independent, shared or partitioned slices on a common infrastructure platform.
- *Dynamic multi-service support, multi-tenancy* and the integration means for vertical market players.
- The *separation of functions*, simplifying the provisioning of services, the manageability of networks, and integration and operational challenges especially for supporting communication services.
- The means for Network /Cloud operators/ ISP and infrastructure owners to reduce operations expenditure, allowing programmability and innovation necessary to enrich the offered services, for providing tailored services, and allowing network programmability to OTT providers and other market players without changing the physical infrastructure.
- *Hosting applications*, offering the capability of hosting virtualized versions of network functions or applications, including the activation of the necessary monitoring information for those functions.
- *Hosting on-demand 3rd parties*/OTTs, empowering partners (3rd parties / OTTs) to directly make offers to the end customers augmenting operator network or other value creation capabilities.

Slicing itself is not new, it has been considered in the past and it is being progressively included in the 5G standards, as discussed below.

2.1 Historical perspective: Early Definitions of Slicing

The followings are early definitions of slicing:

i) Active / Programmable Networks research: node operating systems & resource control frameworks (1995 -2005) produced the text Programmable Networks for IP Service Deployment [SC1].

ii) Federated Testbed research: Planet Lab USA (2002), PlanetLab EU (2005), OneLab EU (2007), PlanetLab Japan (2005), OpenLab EU (2012).

iii) GENI Slice (2008-): "GENI [SC5] is a shared network testbed, i.e., multiple experimenters may be running multiple experiments at the same time. A GENI slice is:

- The unit of isolation for experiments.
- A container for resources used in an experiment. GENI experimenters add GENI resources (compute resources, network links, etc..) to slices, and run experiments that use these resources.
- A unit of access control. The experimenter that creates a slice can determine which project members have access to the slice, i.e., are members of the slice".

iv) Slice capabilities (2009) [SC2]

- 3 Slices Capabilities: "Resource allocation to virtual infrastructures or slices of virtual infrastructure."; "Dynamic creation and management of virtual infrastructures/slices of virtual infrastructure across diverse resources."; "Dynamic mapping and deployment of a service on a virtual infrastructure/slices of virtual infrastructure."
- 17 Orchestration capabilities.
- 19 Self-functionality mechanisms.
- 14 Self-functionality infrastructure capabilities.

v) Cloud manifest (2009) as defined in the RESERVOIR federated cloud environment [SC3] - The paper is ranked 17 out of approx. 25,000 papers published in the last 7 years in Cloud Computing as far as citation is concerned [i.e. August 2018 number of citation for [SC3] is 900] as stated in the review paper [SC9]. The cloud manifest specifies the structure of the service application in terms of component types that are to be deployed as virtual elements. The manifest also specifies the grouping of components into virtual networks and tiers that form the service applications.

vi) ITU-T Slicing (2011) as defined in [SC6], it is the basic concept of the Network Softwarization. Slicing allows logically isolated network partitions (LINP) with a slice being considered as a unit of programmable resources such as network, computation, and storage.

vii) NGMN Slice capabilities (2016) - consist of 3 layers: 1) Service Instance Layer, 2) Network Slice Instance Layer, and 3) Resource layer.

- The Service Instance Layer represents the services (end-user service or business services), which are to be supported. Each service is represented by a Service Instance. Typically, services can be provided by the network operator or by 3rd parties.
- A Network Slice Instance provides the network characteristics, which are required by a Service Instance. A Network Slice Instance may also be shared across multiple Service Instances provided by the network operator.
- The Network Slice Instance may be composed by none, one or more Sub-network Instances, which may be shared by another Network Slice Instance.

viii) IETF (2017) Network Slicing is defined[1] as managed partitions of physical and/or virtual network and computation resources, network physical/virtual and service functions that can act as an independent instance of a connectivity network and/or as a network cloud. Network resources include connectivity, compute, and storage resources. As such Network Slices considerably transform the networking and servicing sperspective by abstracting, isolating, orchestrating, softwarizing, and separating logical network components from the underlying physical network resources and as such they enhance Internet architecture principles.

ix) GPP TR23.799 Study Item "Network Slicing' 2016.

x) ONF Recommendation TR-526 "Applying SDN architecture to Network Slicing" 2016.

xi) EU 5GPPP

- 15 Large Scale Research projects – all based on Network Slicing (https://5g-ppp.eu) (2015-2018+).

[1] IETF draft (2017) "Network Slicing" Galis., A, Dong., J, Makhijani, K , Bryant, S., Boucadair, M, Martinez-Julia,P. https://tools.ietf.org/html/draft-gdmb-netslices-intro-and-ps-01

- White Papers on 5G Architecture centerd on network slicing (mark 1 - (2016) [SC7]) (mark 2 - (2018) [SC8].

xii) Additional characteristics, standard and research activities on Infrastructure slicing and references are presented in [SC4].

Conclusions on the Emerging Slicing Concepts and Early Definitions of Slicing

After introducing the emerging slicing concepts, we observe that early definitions of slicing have diverse scope as well as major differences in the enabling technological underpinnings (e.g., SDN, NFV, etc.). Ongoing standardization efforts around slicing are closer to recent broadly scoped definitions of slicing and the key characteristics of cloud network slicing. However, standardization activities are too siloed off from each other and present fundamental divergences on their approach (i.e. where slicing is done and functional scope). We observe that existing approaches do not fully attend the current and foreseeable needs of cloud network slicing in its broadest capabilities.

One of the limitations we observed is related to the recurrent peer-to-peer interaction between different providers / domains sharing resources for the slicing. We expect the implementation of an alternative approach where sliced resources are directly accessible and attached to an end-to-end Slice by a provider initiating the Slice creation workflow. This will overcome the need of peer-to-peer interactions among different providers after the end-to-end slice has been put in place and becomes the driving objective of the NECOS project.

To clarify how NECOS aims at filling gaps towards the realization of network cloud slicing, in the following we survey the state of the art[2] in standardization and research projects and conclude with a summarizing discussion on which key features of the cloud network slicing are being covered.

2.2 ITU-T Slicing

References:

- ITU-T Y.3011- http://www.itu.int/rec/T-REC-Y.3001-201105-I
- ITU-T IMT-2020 - https://www.itu.int/en/publications/Documents/tsb/2017-IMT2020-deliverables/mobile/index.html#p=49

According to ITU-T Y.3011 (2011) slicing allows logically isolated network partitions (LINP) with a slice being considered as a unit of programmable resources such as network, computation and storage. LINPs are isolated from each other, and when combined with programmability in virtual resources, users of LINPs can program the virtual resources on above the virtualization layer. In other words, each LINP can provide the corresponding users with services similar to those provided by traditional networks without network virtualization. The users of LINPs are not limited to the users of services or applications, but can include service providers. For example, a service provider can lease a LINP and can provide emerging services or technologies such as cloud computing service. The service providers can realize the emerging services as if they own a dedicated physical network. In order to facilitate the deployment of network virtualization, it is necessary to provide control and management procedures such as creating, monitoring, and measuring the status of LINPs.

[2] For a detailed state-of-the art review of slicing (history, terms & concepts, use cases, SDOs, research projects, remaining challenges, references), interested readers are pointed to the Network Slicing Tutorial at IEEE CNSM, Rome, 5-9 November 2018, publicly available at http://cnsm-conf.org/2018/files/CNSM18_SlicingTutorial_AlexGalis_5-10-2018.pdf

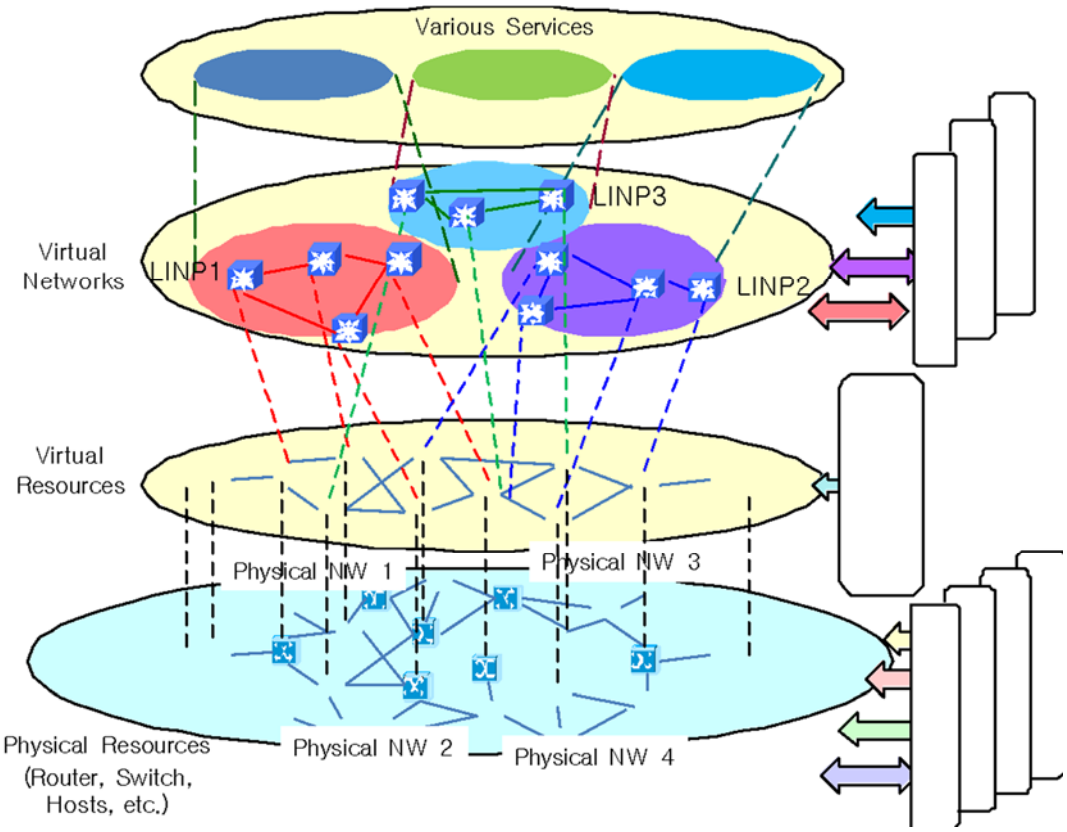

Figure 1. Conceptual architecture of network virtualization

Figure 1 represents the conceptual architecture of network virtualization, which consists of LINPs over physical resources supporting network virtualization.

A single physical resource can be shared among multiple virtual resources and each LINP consists of multiple virtual resources. Each LINP is managed by individual LINP manager.

2.3 NGMN Slicing

References:

- White Paper on 5G (2016) - https://www.ngmn.org/5g-white-paper/5g-white-paper.html
- NGMN Alliance "Description of Network Slicing Concept" - https://www.ngmn.org/fileadmin/user_upload/160113_Network_Slicing_v1_0.pdf.

According to NGMN slicing consists of 3 layers (1) Service Instance Layer, (2) Network Slice Instance Layer, and (3) Resource layer:

- The Service Instance Layer represents the services (end-user service or business services) which are to be supported. Each service is represented by a Service Instance. Typically, services can be provided by the network operator or by 3rd parties.
- A Network Slice Instance provides the network characteristics which are required by a Service Instance. A Network Slice Instance may also be shared across multiple Service Instances provided by the network operator.
- The Network Slice Instance may be composed by none, one or more Sub-network Instances, which may be shared by another Network Slice Instance.

The Network Slice Instance may be composed by none, one or more Sub-network Instances, which may be shared by another Network Slice Instance. Similarly, the Sub-network Blueprint is used to create a Sub-network Instance to form a set of Network Functions, which run on the physical/logical resources.

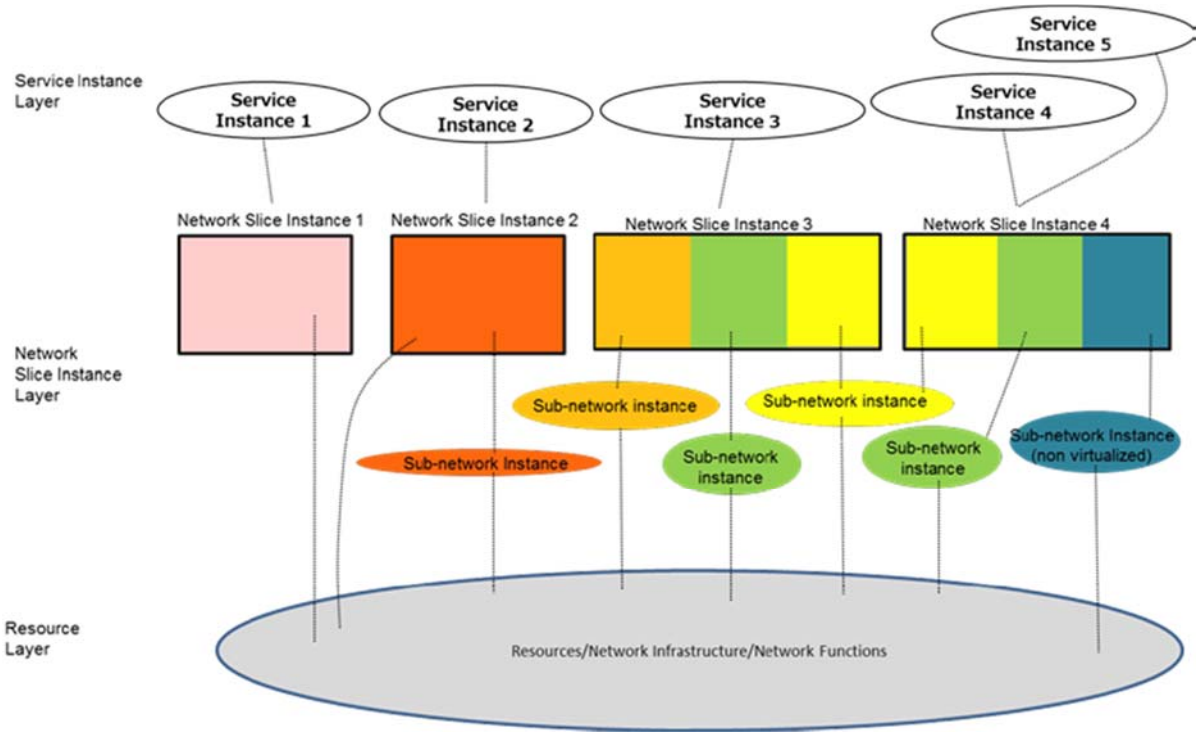

Figure 2. Network slicing conceptual outline

2.4 IETF Slicing

References: (including work in progress):

- IETF draft (2017) "Network Slicing" Galis., A, Dong., J, Makhijani, K , Bryant, S., Boucadair, M, Martinez-Julia,P. https://tools.ietf.org/html/draft-gdmb-netslices-intro-and-ps-01
- NetSlices Architecture draft-geng-netslices-architecture-02
- General Requirements for Network Slicing: https://www.ietf.org/id/draft-homma-slice-provision-models-00.txt
- Network Slicing - Revised Problem Statement draft-galis-netslices-revised-problem-statement-03
- NetSlices Management Architecture draft-geng-coms-architecture-01
- NetSlices Use Cases draft-netslices-usecases-01
- NetSlices Management Use cases draft-qiang-coms-use-cases-00
- NetSlices Information Model draft-qiang-coms-netslicing-information-model-02
- Autonomic NetSlicing draft-galis-anima-autonomic-slice-networking-04
- NetSlices Interconnections https://tools.ietf.org/html/draft-defoy-coms-subnet-interconnection-03

According to the IETF network slicing architecture the following terms are defined:

- Resource Slice - A grouping of physical or virtual (network, compute, storage) resources. It inherits the characteristics of the resources which are also bound to the capability of the resource. A resource slice could be one of the components of Network Slice, however on its own does not represent fully a Network Slice.

- Network Slice - A Network slice is a managed group of subsets of resources, network functions / network virtual functions at the data, control, management/orchestration planes and services at a given time. Network slice is programmable and has the ability to expose its capabilities. The behaviour of the network slice is realized via network slice instance(s).
- End-to-end Network Slice - A cross-domain network slice which may consist of access network (fixed or cellular), transport network, (mobile) core network and etc. End-to-end network slice can be customized according to the requirements of network slice tenants
- Network Slice Instance - An activated network slice. It is created based on network template. A set of managed run-time network functions, and resources to run these network functions, forming a complete instantiated logical network to meet certain network characteristics required by the service instance(s). It provides the network characteristics that are required by a service instance. A network slice instance may also be shared across multiple service instances provided by the network operator.

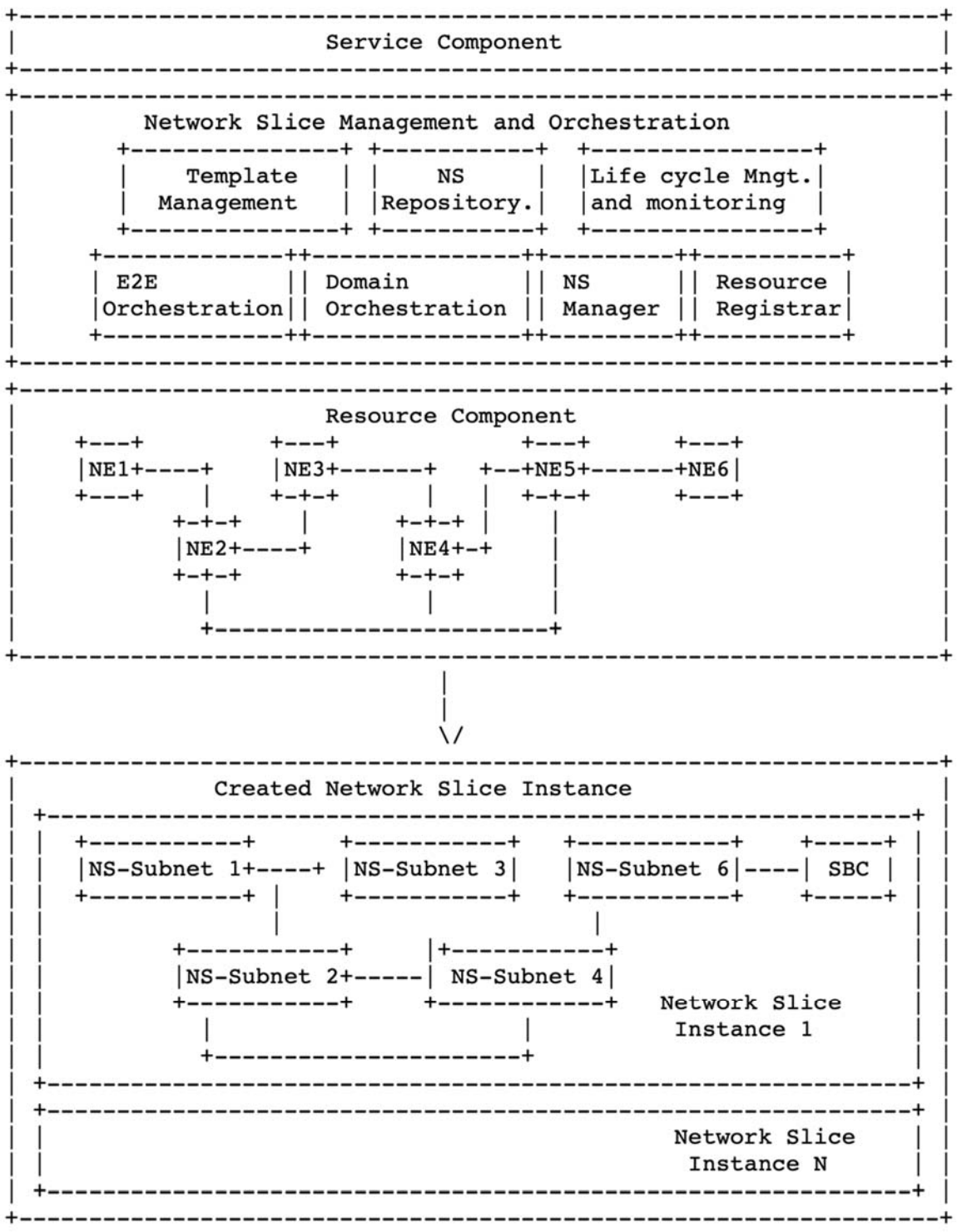

Figure 3. IETF Network slicing architecture

2.5 3GPP Slicing

References: 3GPP: www.3gpp.org/ Published & Work In progress on slicing includes:

- 3GPP SA2 - Study on Architecture for Next Generation System /Network slice related functionality (3GPP TR 23.799)

- 3GPP SA2 - System Architecture for the 5G System /Network slice related functionality (3GPP TS 23.501)
- 3GPP SA2 - Procedures for the 5G System: Procedures and flows of the architectural elements/ Network slice related procedures (3GPP TS 23.502)
- 3GPP SA3 - Study on the security aspects of the next generation system/ Network slice related security (3GPP TR 33.899)
- 3GPP SA5 - Study on management and orchestration of network slicing/ Network slice management (3GPP TR 28.801)
- 3GPP SA5 - Provisioning of network slicing for 5G networks and services: Detailed specification of network slice provisioning/ Network slice management (3GPP TS 28.531)
- 3GPP SA5 - Management of network slicing in mobile networks - concepts, use cases and requirements (3GPPTS 28.530)
- 3GPP TR 28.801 - Study on management and orchestration of network slicing for next generation network

According to 3GPP TR 28.801 the network slice concept includes the following aspects:

1. Completeness of an NSI (network slice instance): An NSI is complete in the sense that it includes all functionalities and resources necessary to support certain set of communication services thus serving certain business purpose.
2. Components of an NSI:

   The NSI contains NFs (e.g. belonging to AN and CN). If the NFs are interconnected, the 3GPP management system contains the information relevant to the connections between these NFs such as topology of connections, individual link requirements (e.g. QoS attributes), etc. For the part of the TN (Transport Network) supporting connectivity between the NFs, the 3GPP management system provides link requirements (e.g. topology, QoS attributes) to the management system that handles the part of the TN supporting connectivity between the NFs.
3. Resources used by the NSI: The NSI is realized via the required physical and logical resources.
4. Network Slice Template: The network slice is described by a Network Slice Template (NST). The NSI is created using the NST and instance-specific information.
5. NSI policies and configurations:

   Instance-specific policies and configurations are required when creating an NSI Network characteristics examples are ultra-low-latency, ultra-reliability, etc. NSI contains a Core Network part and an Access Network part.
6. Isolation of NSIs: A NSI may be fully or partly, logically and/or physically, isolated from another NSI

The network slice stakeholders involved in 3GPP network slice management are depicted in the following figure:

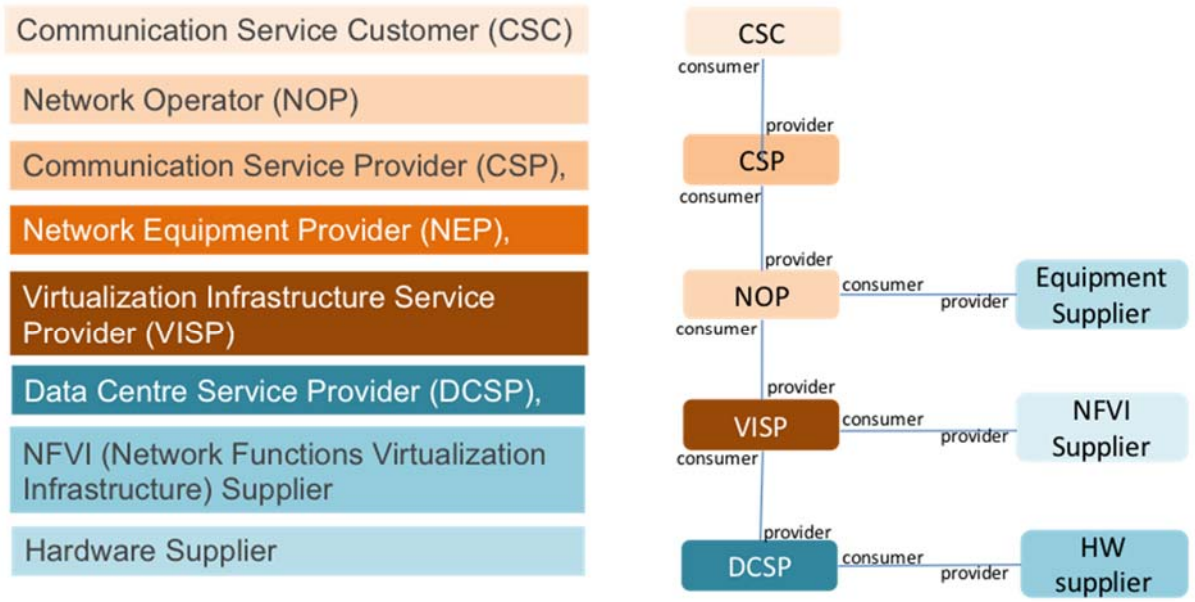

Figure 4. 3GPP network slice stakeholders

There is also a collaboration between IETF and 3GPP, where it has been suggested that the best approach is to ensure that the 3GPP engineers are involved in the IETF work they are interested in, and that 3GPP states clearly what their requirements (rather than solutions) are. IETF also noted that the work in 3GPP is ongoing.

2.6 ETSI Slicing

References:

- Network Operator Perspectives on NFV priorities for 5G- https://portal.etsi.org/NFV/NFV_White_Paper_5G.pdf
- Report on Net Slicing Support with ETSI NFV Architecture Framework http://www.etsi.org/deliver/etsi_gr/NFV-EVE/001_099/012/03.01.01_60/gr_NFV-EVE012v030101p.pdf
- "E2E Network Slicing Reference Framework and Information Model" - Kiran, Makhijani- Huawei Technologies, Kevin Smith-Vodafone John Grant - Ninetiles Alex Galis- UCL, Xavier Defoy- Interdigital - approved & published in October 2018 by ETSI as a standard specification document- https://www.etsi.org/deliver/etsi_gr/NGP/001_099/011/01.01.01_60/gr_ngp011v010101p.pdf

ETSI's E2E Network Slicing identifies a next-gen network slicing (NGNS) framework defined here is a generalized architecture that would allow different network service providers to coordinate and concurrently operate different services as active network slices, including slicing design principle (service oriented approach, slice abstraction, slice reusability, slice autonomy), an information model, network slice functions specification and slice enablement.

The NS methods are aimed at providing custom design of networks suitable for a specific use case (vertical market). Such methods need to be able to translate a service requirement into normalized description of resources across different type of network domains based on NGMN's description of network slicing. The 3-layer approach is as follows:

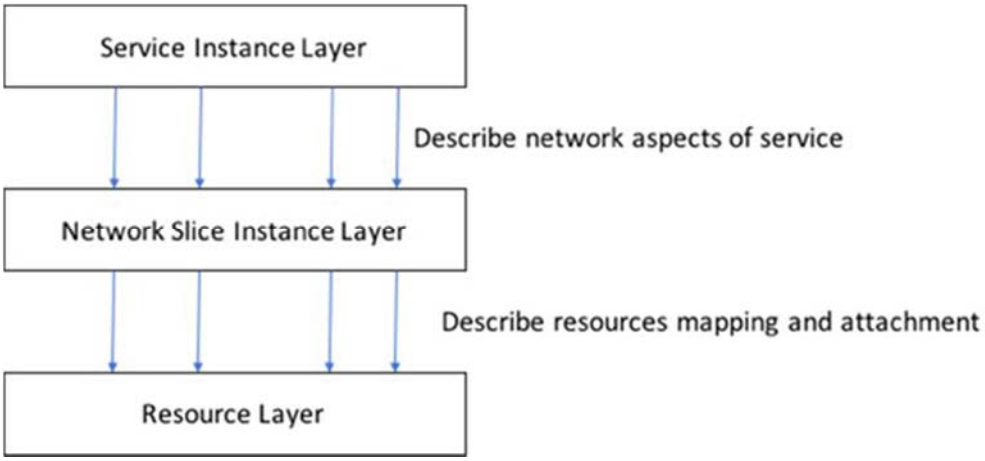

Figure 5. 3-layer network slice and service concept

According to ETSI NGP there are three key areas of consideration for the NS architecture:

1. Service description (corresponds to service instance layer): A sketch of services instantiated, independent of any technology or underlying control plane.
2. Network slice to abstract resource mapping (corresponds to network slice instance layer).
3. Resource allocation (across different networks).

2.7 ONF Slicing

Reference: TR-526 Applying_SDN_Architecture_to_5G_Slicing_TR-526.pdf

The ONF Slicing adopts the NGMN terminology of network slicing.

Applying slicing to the SDN architecture, the client context provides the complete abstract set of resources (as Resource Group) and supporting control logic that constitutes a slice, including the complete collection of related client service attributes. As such, a 5G slice is comparable to, if not the same as, an SDN client context, isolated by the controller's virtualization and client policy functions and continuously optimized by the orchestration and global policy functions. Going a step further, a 5G controller is an SDN controller, or vice versa. Both 5G and SDN controller manage-control combinations of all relevant network resources / functions / assets required to serve client- specific purposes. The client context also offers to the client functions to manage-control the slice resources, including OAM related-functions, as visible by / available to the client according to administrative policy.

Recursion in the SDN architecture allows for a high-level client context to be virtualized and orchestrated over a number of lower-level resource groups, spanning geographic, business and technology boundaries as needed. Each lower-level resource group may itself be exposed by the client context of a lower-level SDN controller. The SDN architecture thereby naturally supports a recursive composition of slices.

The SDN controller, at the center of a feedback loop and acting autonomously according to administratively configured policies, enables dynamic allocation, modification and optimization of resource usage. The resources provided to the controller's clients are virtualized (abstracted views and services) from the underlying resources and presented to slices through resource groups.

Figure 6 shows a mapping of the SDN architecture, terminology and abstractions to 5G slicing.

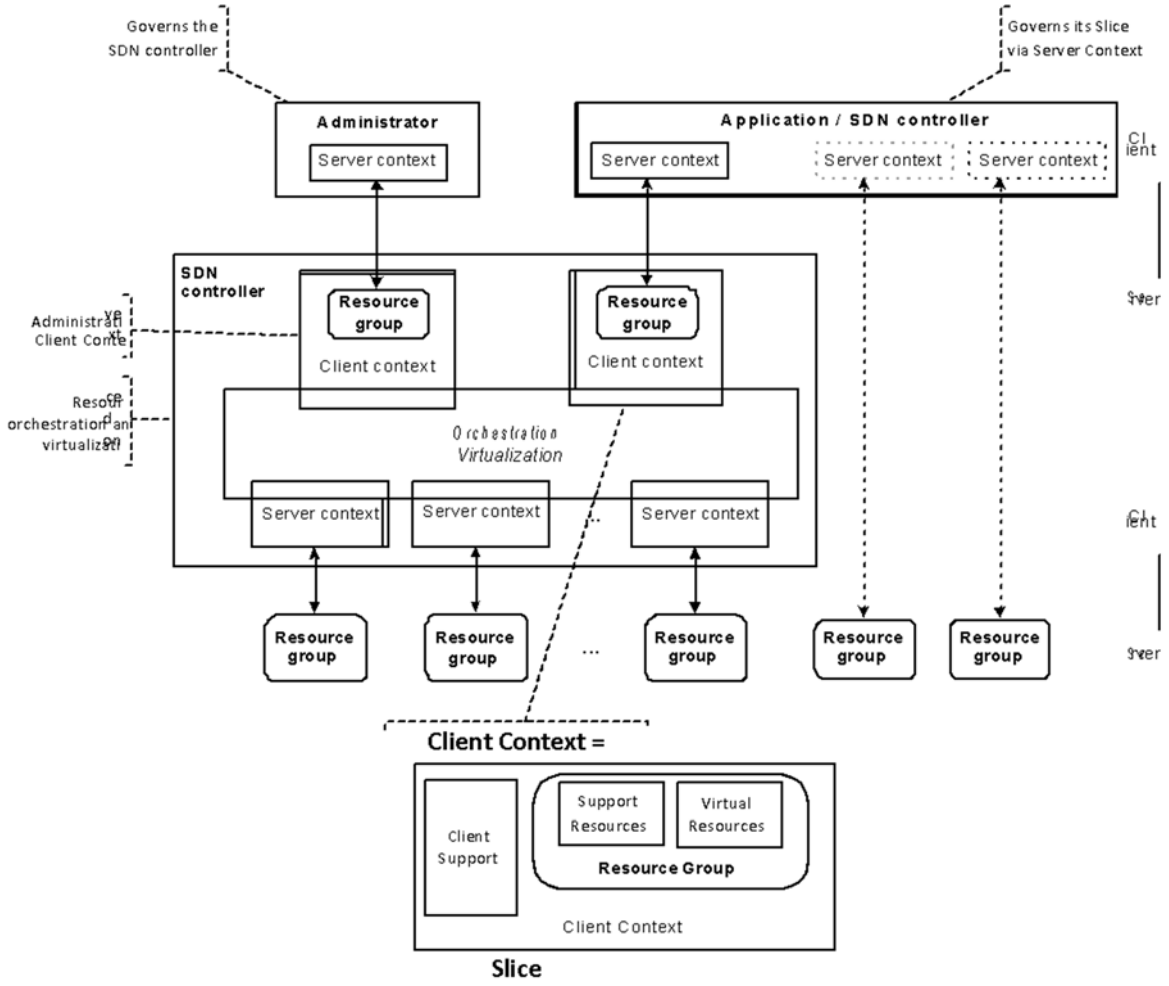

Figure 6. ONF slicing abstractions

2.8 EU 5GPPP Projects

In this section we present several 5GPPP projects which were selected as they have had design and implementation facets that are focussed on slicing. These include: 5GEx; 5G SONATA; 5G NORMA; 5G-Transformer; 5G-PAGODA; and 5G-SLICENET.

2.8.1 5GEX EU Project

Reference: http://www.5gex.eu/

5GEx focused on the transition from dedicated physical networks with dedicated control and dedicated services and resources for different applications to a "network factory" where resources and network functions are traded and provisioned – a new infrastructure and services "manufactured by SW" as depicted in Figure 7.

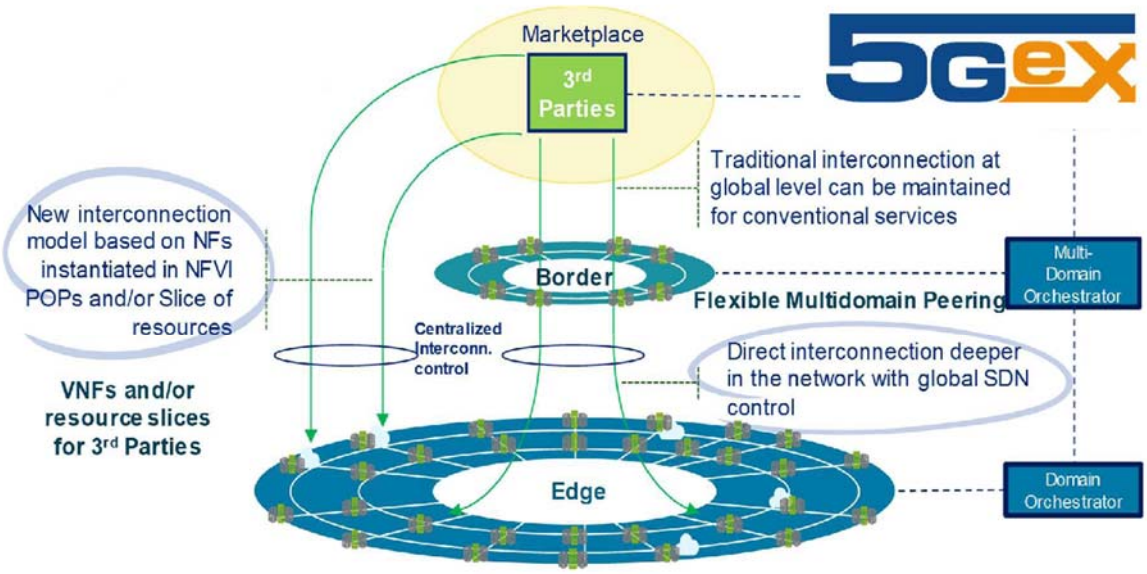

Figure 7. 5GEX network softwarization process - Infrastructure and services manufactured by SW

In 5GEx a Slicer management function is design with the view to enrich the Resource Orchestrator functionality with multi-tenancy resource slicing and generic lifecycle VNF management functionality. Additionally, the Slicer function coordinates and integrates multi-vendor / 3rd party specific lifecycle VNF management functions.

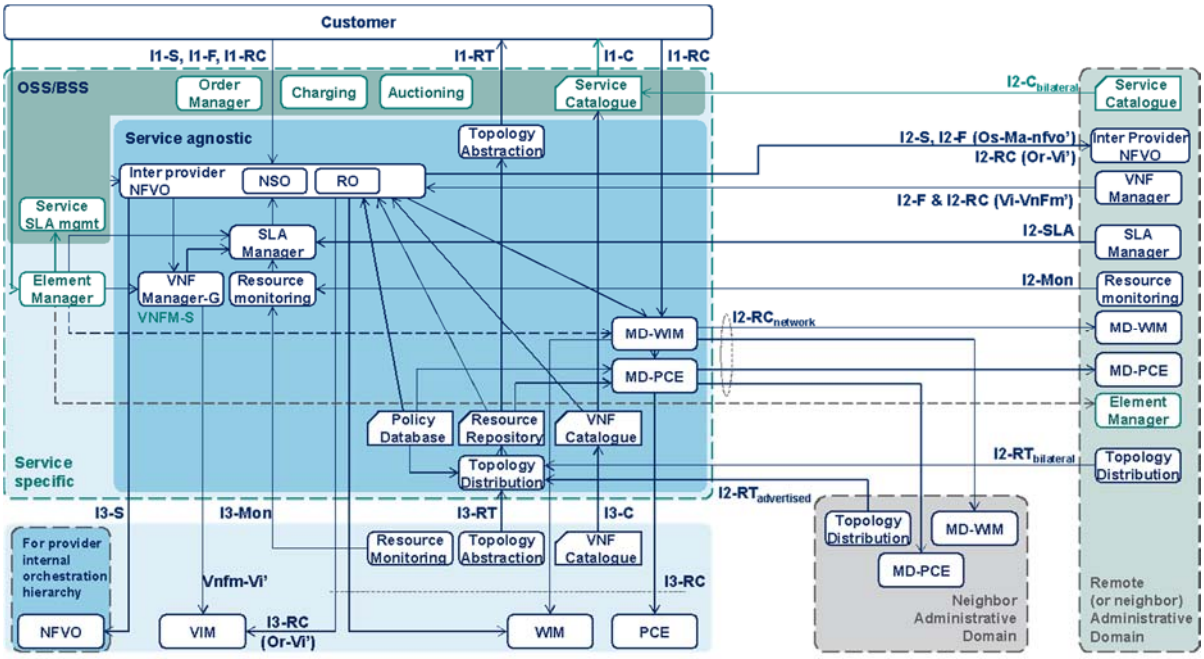

Figure 8. Functional model of 5GEX multi-domain orchestration

2.8.2 5G-SONATA EU Project

Reference: http://sonata-nfv.eu

5G SONATA focussed on flexible programmability of software networks and the optimization of their deployments.

As far as SONATA is concern a slice is an aggregated set of resources that can be used in the context of an end-to-end networked service comprised of virtual network functions. Slices are composed of multiple resources which are isolated from other slices and allows logically isolated network partitions, with a slice being considered as the basic unit of programmability using network, computation and storage.

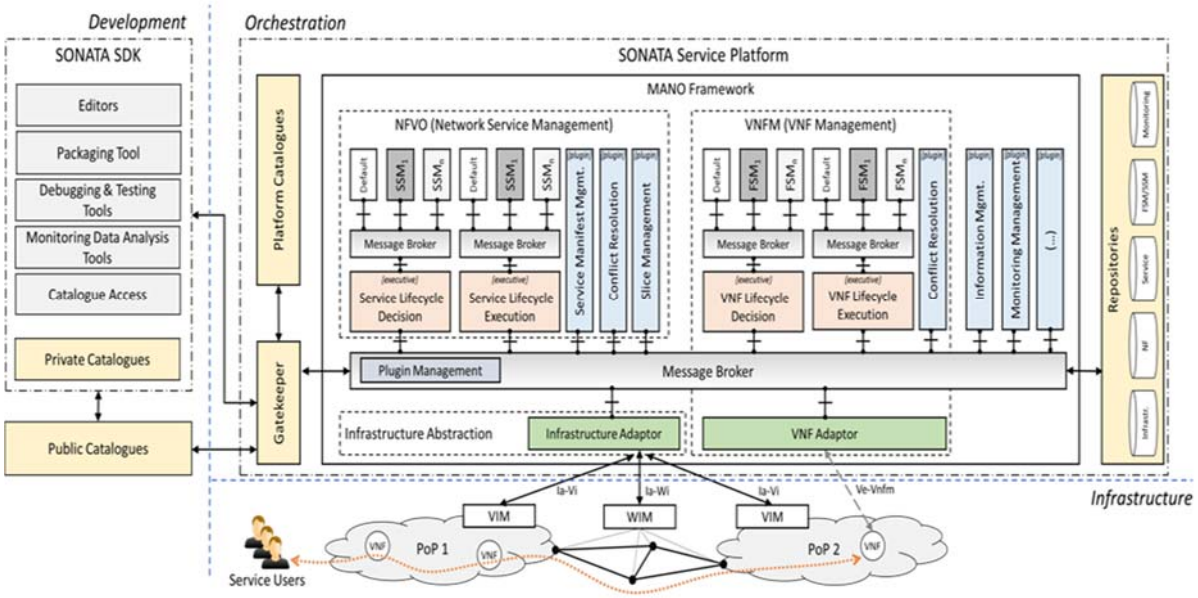

Figure 9. SONATA high level architecture

2.8.3 5G-NORMA EU Project

Reference: http://www.it.uc3m.es/wnl/5gnorma/

The 5G-NORMA focus is on: i) Adaptive (de)composition and allocation of NFs Joint; ii) Optimization of RAN and Core Network and iii) SW-defined Mobile Control.

This involved the design for Service management; Mapping of customer-facing services and procedures to resource-facing services and procedures; Network slicing (Service and Resource); Orchestration; Inter-slice and intra-slice and Network programmability as depicted in Figure 10.

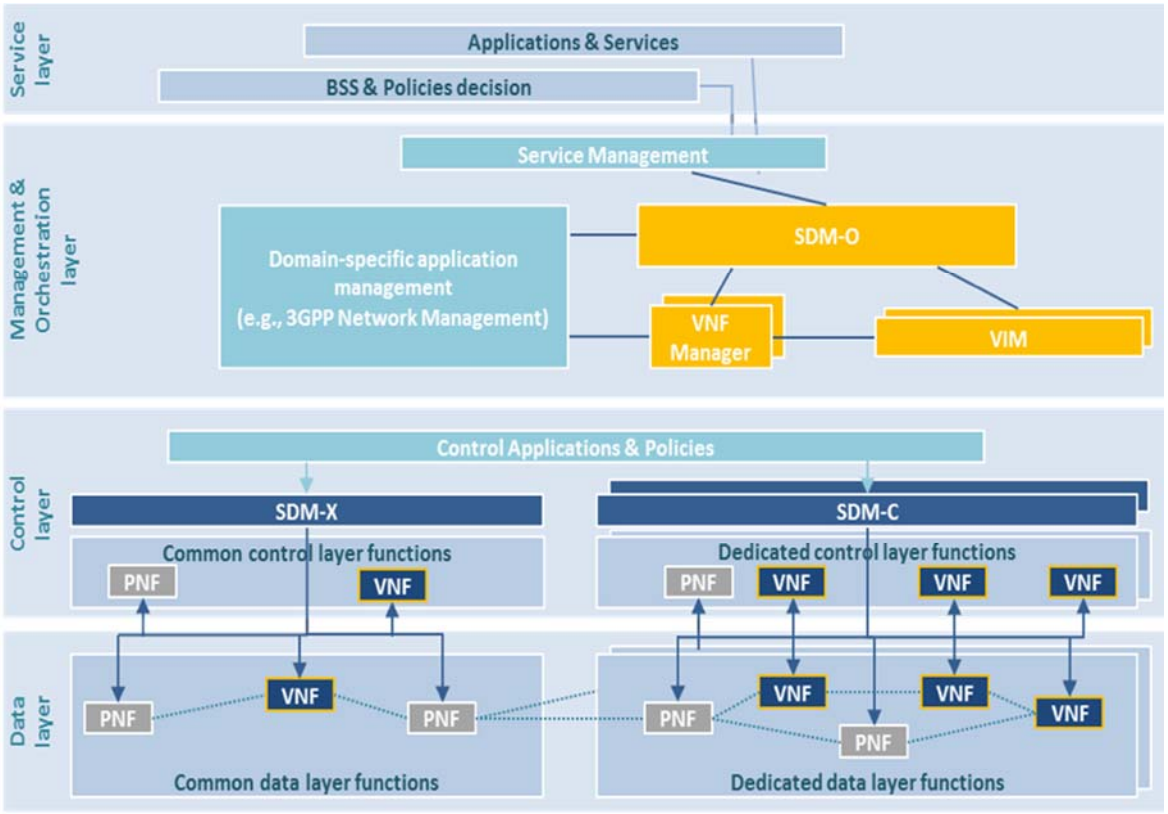

Figure 10. 5G NORMA functional architecture

2.8.4 5G-TRANSFORMER EU Project

Reference: http://5g-transformer.eu

5G-Transformer focuses on (i) Vertical Slicer as the logical entry point for verticals to support the creation of their respective transport slices; (ii) Sefrvice Orchestrator to orchestrate the federation of transport networking and computing resources from multiple domains and manage their allocation to slices, and (iii) Mobile Transport and Computing Platform or integrated fronthaul and backhaul networks as depicted in Figure 11.

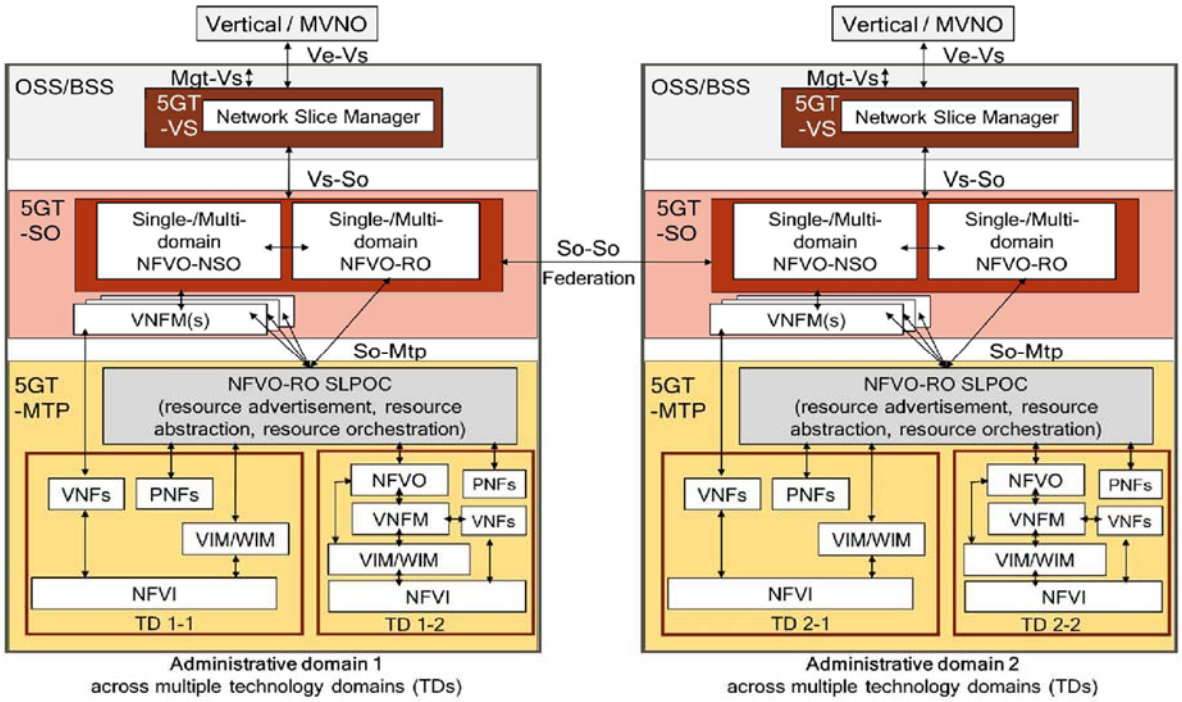

Figure 11. 5G Transformer slice architecture

2.8.5 5G-PAGODA EU Project

Reference: https://5g-pagoda.aalto.fi

5G PAGODA proposal is based on: (i) scalable 5G slicing architecture; (ii) extending the current NFV architecture towards support of different specialized network slices composed of multi-vendor virtualized network functions scalable 5G slicing architecture, and (iii) extending the current NFV architecture towards support of different specialized network slices composed of multi-vendor virtualized network functions as depicted in Figure 12.

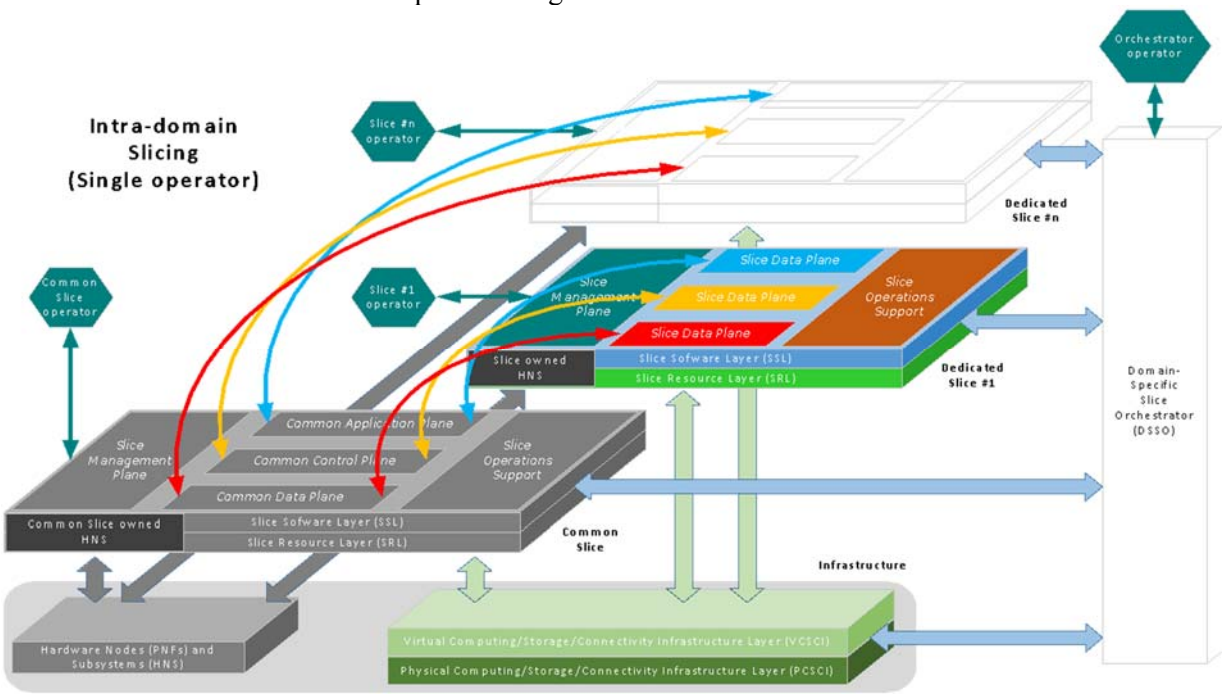

Figure 12. 5G PAGODA intra-domain slicing architecture

2.8.6 5G-SLICENET EU Project

Reference: https://slicenet.eu
5G-SLICENET is focusing on: (i) End-to-End Cognitive Network Slicing and (ii) Slice Management Framework in Virtualised Multi-Domain, Multi-Tenant 5G Networks as depicted in Figure 13.

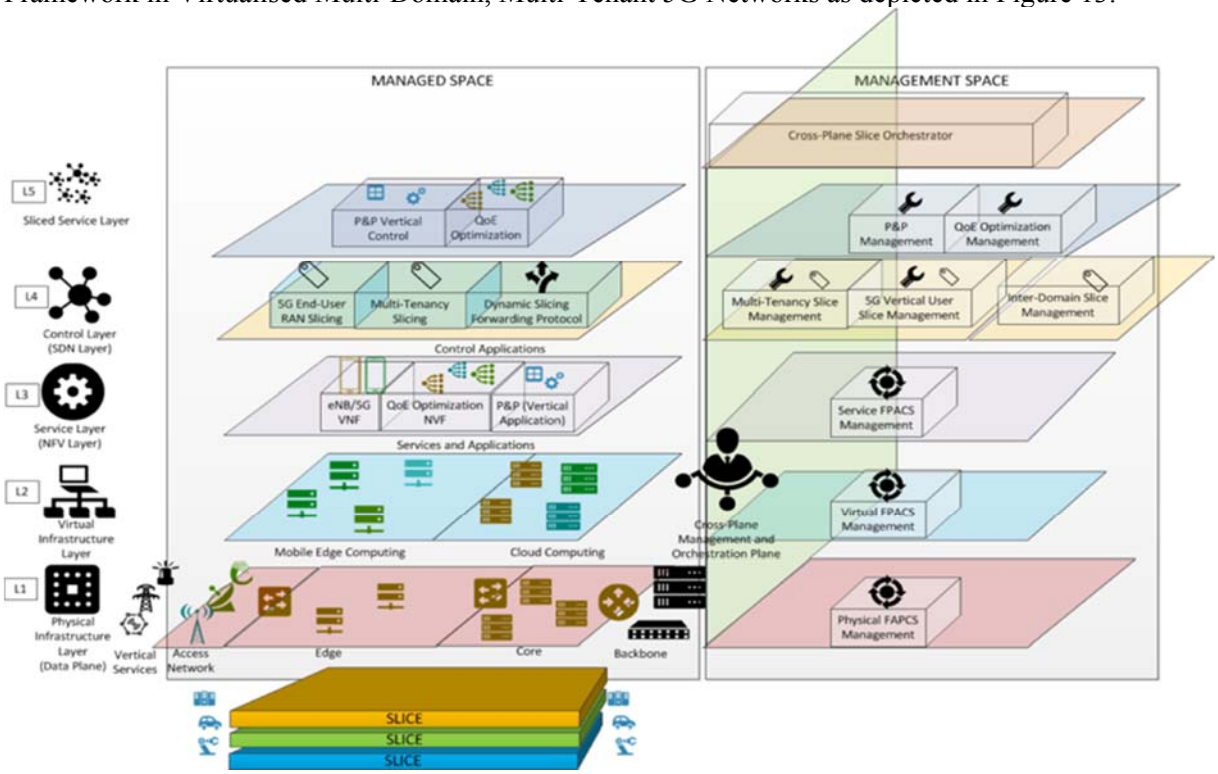

Figure 13. 5G SLICENET high level slicing architecture

2.9 Summary of Network Slicing Characteristics

After reviewing a set of relevant network slicing activities in standardization and research projects, it becomes clear that there is a lack of common terminology and consistent concepts to clearly characterize the scope and approach of each work embracing network slicing principles. As an effort to provide some qualitative comparison on how each work is approaching network slicing, and also to better position these work activities within the NECOS project scope, we have identified a (non exhaustive) list of 10 key characteristics at a high and common level, sufficient to provide a comprehensive snapshot on the state of affairs towards network slicing. The distinguishing characteristics are as follows:

- Connectivity Resource only Slicing
- Connectivity Service Slicing
- Network Cloud Slicing
- E2E Slicing
- Multi-domain Slicing
- Uniform Lifecycle Management
- Tenant Slice Management
- Slices as a Service
- VIM/WIM on Demand
- Marketplace for Slices

While the set of selected initiatives and projects is representative enough for our main goal of positioning NECOS with regard to relevant related work, we recognize that more projects could be brought into the picture, especially as time passes, since the support of some type of slicing features is increasingly becoming the norm of any network and cloud oriented initiative. Certainly, further characteristics could be identified in the future for a more fine-grain and scoped discussion.

The following table summarises how the key slice characteristics (i.e., are being addressed) in the first *blue* column are mapped to the different initiatives in *green* per column. While we opted for a binary check on each characteristic, we recognise that finer-grain quantification and qualification could be done especially for those characteristics ticked as positive by different works.

We can observe that some characteristics, namely (i) Connectivity Resource only Slicing, (ii) Connectivity Service Slicing, (iii) Uniform Lifecycle Management, and (iv) Tenant Slice Management, have been majorly more worked out by different initiatives. This is sensible considering the initial standpoint of *network slicing* was from network connectivity-oriented initiatives and the common management requirements of resources and tenant slices.

In contrast, we note that few works address cloud slicing, or jointly address cloud and network slicing, or consider end-to-end and multi-administrative domain scenarios, or embrace the Slice-as-a-Service paradigm. NECOS not only addresses those key characteristics, but in addition goes beyond the state of the art by considering a number of unique features, namely the VIM on-demand and WIM on-demand slicing models, and the Marketplace approach, which have not been considered by other slicing approaches. These will help deliver a more flexible concept of slicing which goes beyond the peer-to-peer orchestration and management interactions on which most of the above initiatives are based. Our proposed architecture will start from these concepts to introduce and design the blocks that will allow the realisation of a slicing approach that will cover the highlighted characteristics.

Table 1. Comparison of slicing initiatives and projects with NECOS

	Initiative						**EU 5GPPP Project**						
	ITU-T Slicing	*NGMN Slicing*	*IETF Slicing*	*3GPP Slicing*	*ETSI Slicing*	*ONF Slicing*	*5GEX Slicing*	*5G-SONATA Slicing*	*5G_NORMA Slicing*	*5G-Transformer Slicing*	*5G-PAGODA Slicing*	*5G-Slicenet Slicing*	***NECOS Slicing***
Slicing Characteristics													
Connectivity Resource only Slicing	✓	✗	✗	✓	✗	✓	✓	✗	✗	✓	✓	✓	✓
Connectivity Service Slicing	✗	✓	✓	✗	✓	✗	✗	✓	✓	✗	✗	✗	✓
Network Cloud Slicing	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
E2E Slicing	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓
Multi-domain Slicing	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓
Uniform Lifecycle Management	✗	✗	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓
Tenant Slice Management	✗	✗	✓	✓	✓	✗	✓	✓	✗	✓	✓	✗	✓
Slices as a Service	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓
VIM/WIM on Demand	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

Marketplace for Slices	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

2.10 NECOS view on slicing: Key terms and definitions

Network technologies are making rapid progress to support new 5G communications (e.g., URLLC - Ultra Reliable and Low Latency Communication, mMTC - massive Machine Type Communication, eMBB - enhanced Mobile Broadband, and UDN – Ultra Dense Networking) and IoT trends. Emerging vertical industry applications, such as autonomous/cooperative driving, drone surveillance, remote sensing, and data analytic, are causing the conventional cloud computing model to evolve towards edge computing, by utilizing these advanced communications so that obtaining ultra-low latency, high broadband, and customized data transport service. In the light of this observation and the native integration of network and cloud computing domains, investigating into the interaction between ***cloud*** computing and advanced ***net*****working** is a goal of the NECOS project.

One of the critical driving issues for such integration and interworking is the simple fact that *is extremely inefficient and expensive to build a separate computing & networking infrastructure for each and/or new service.*

The followings are key concepts utilized in the NECOS project architecture and system design.

1. Key Roles:

Resource Provider –- It owns the physical resources and infrastructure (network/ cloud/ datacenter) and provides / leases them to operators.

Slice Provider– A slice provider, typically a telecommunication service provider, is the owner or tenant of the infrastructures from which cloud network slices can be created.

Slice Tenant – A slice tenant is the user of a specific network/cloud/datacenter slice, in which customized services are hosted. Infrastructure slice tenants can make requests for the creation of new infrastructure slice through a service model.

2. Key Concepts:

Cloud Network Slice –- A set of infrastructures (network, cloud, data center) components/network functions, infrastructure resources (i.e., managed connectivity, compute, storage resources) and service functions that have attributes specifically designed to meet the needs of an industry vertical or a service. As such a Cloud Network Slice is a managed group of subsets of resources, network functions/network virtual functions at the data, control, management/orchestration, and service planes at any given time. The behaviour of the Cloud Network Slice is realized via network slice instances (i.e., activated slices, dynamically and non-disruptively re-provisioned). The Cloud Network Slice key concepts are provided below:

- *A cloud network slice supports at least one type of service.*
- *A cloud network slice may consist of cross-domain components from separate domains in the same or different administrations, or components applicable to the infrastructure.*
- *A resource-only partition is one of the components of a Cloud Network Slice, however on its own does not fully represent a Network Slice.*
- *A collection of partitions from separate domains is combined and aggregated to form a cloud network slice.*
- *Underlays / overlays supporting all services equally (with 'best effort" support) are not fully representing a Network Slice.*

Infrastructure Slicing –- A management mechanism that an Infrastructure Slice Provider can use to allocate dedicated infrastructure resources and service functions to Network Slice Tenant. Broadly, partition strategies can be classified into three categories:

- Physical separation, such as dedicated backbones or dedicated data centers. However, this strategy is not cost efficient.
- Underlays/overlays supporting all services equally ('best effort" support), such as underlays / overlays, in the form of VPN as overlay solution. These solutions are neither flexible nor agile.
- Slicing, through cloud network resources allocation, where dedicated resources per customer/service ensure both isolation and customization on top of the same infrastructure.

2.11 Slicing, sharing and partitioning of resources

- From **a business point of view, a Cloud Network slice** includes *a combination of all the relevant network and compute resources, functions, and assets required to fulfil a specific business case or service.*
- From the **infrastructure point of view, infrastructure slice instances** require *the partitioning and assignment of a set of resources that can be used in an isolated, disjunctive or non-disjunctive manner for that slice.*
- From the **tenant point of view, infrastructure slice instance provides** different capabilities, specifically in terms of their management and control capabilities, and how much of them the network service provider hands over to the slice tenant. As such, there are two types of slices:
  (1) **Internal slices**, understood *as the partitions used for internal services of the provider, retaining full control and management of them.*
  (2) **External slices**, being those *partitions hosting customer services, appearing to the customer as dedicated networks/clouds/data centers.*
- From the **management plane point of view, infrastructure slices** refer to the managed fully functional *dynamically created partitions of physical and/or virtual network resources, network physical/virtual and service functions that can act as an independent instance of a connectivity network and/or as a network cloud.* Infrastructure resources include connectivity, compute, and storage resources.
- From the **date plane point of view, infrastructure slices** refer to *dynamically created partitions of network forwarding devices with guarantees for isolation, customization and security*.

2.12 Slicing operational modes

After reviewing the state of the art on slicing, including the academic literature, the very much fragmented standardization landscape, as well as the enabling technologies and early realizations of slicing concepts, one relevant observation of the NECOS project is the lack of a unique approach (even at the conceptual level) to afford Cloud Network slicing. We understand that the quest of providing sliced services by logically partitioning resources of a (multi-domain) software-defined infrastructure can be pursued through different slicing approaches depending at which level (and entry-point) of the stack slicing is performed. These different options are what we call *slicing operation modes* (*modus operandi*).

The choice of the slicing operational mode has a critical impact on the amount and type of components that need to be slice-aware, i.e., requiring software changes to support the slicing functionality, as well as the degrees of freedom on implementation choices, the isolation properties and the control and management responsibilities of both Slice Providers and Slice Tenants.

Figure 1 presents a high-level layered view of software-defined infrastructure organized in the three planes: (i) the *infrastructure* consisting of resources of different types (e.g., computing, networking), (ii) the *control and management* realm encompassing resource management entities (e.g., VIM, WIM, VNFM), control and orchestration functions, and (iii) the *business* plane featuring the applications and functions of an end-to-end service composed from an arbitrary number of network-related and application-specific end- or middle-point functions. Monitoring functionalities are assumed on each plane.

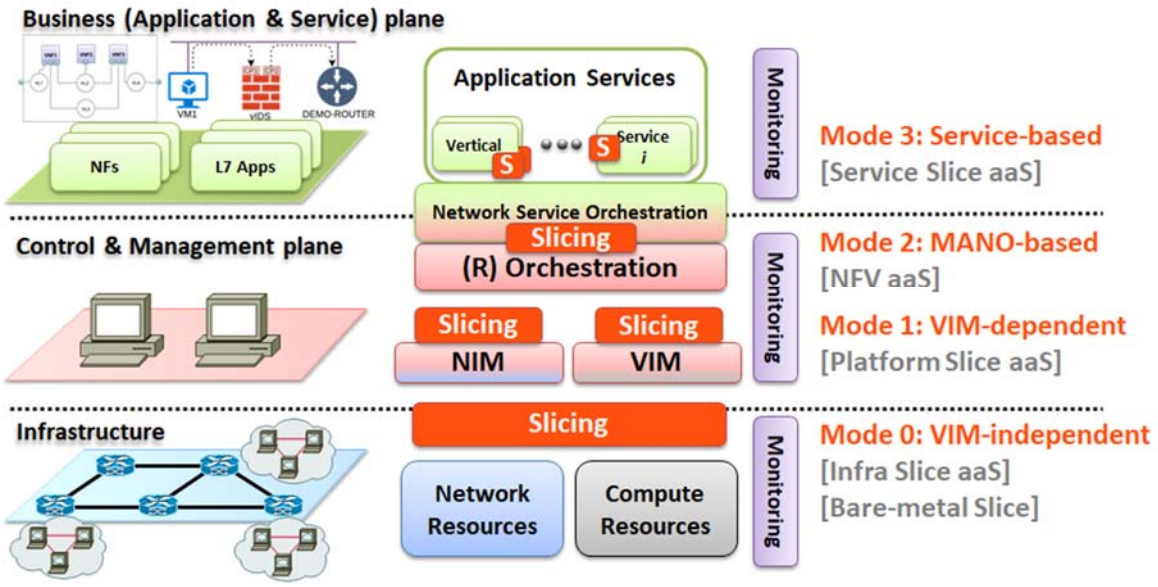

Figure 14: Candidate alternative slicing approaches

As illustrated by the dark orange boxes, labelled with Slicing in Figure 1, the logical partition of resources and their control and management functions to provide the slices can be realized at different architectural points. One fundamental question arises when considering the different plausible alternatives, which revolves around the need for changes. So, do we change all of the software to deal with slices, or do lower layer abstractions suffice to present a slice that all software just runs on?

Slicing at lower layers means that upper layers, such as VIMs and Orchestrators do not need to know about slicing. If a slice is presented to them, they can carry on working with no change or minimal change. If slicing is done otherwise, then all the main software elements need to be updated and adjusted to know about slices, i.e., all of the APIs, the modules, and internal function paths need to be adjusted and adapted to factor in slices. Hence, there are inherent trade-offs when selecting one or the other slicing operational mode. Ultimately, the actual decision on which slicing approach will depend on multiple criteria on key aspects of the use case scenario under consideration, dominated by the technical footprint of the provider, and the technical abilities and technological choices of the tenants and interest in opting for more or less control and responsibilities including customized software components. In the following, we discuss the key aspects of each slicing approach before we conclude the best fits to the goals and scope of NECOS.

These points show how candidate approaches (slicing modes) and the architecture meet, discussing the main characteristics, the SW dependencies and impact of slice-ready APIs:

- **Mode 0 - VIM-independent slicing:** the VIM on-demand model with DC slicing that allows direct inter-domain orchestrator to VIM interaction using a specific allocated VIM. Multi-tenancy happens at the lower resource layers. Being a VIM-independent approach, tenants can choose and have direct control on the VIMs, which can be as lightweight as desired. However, the freedom of choice and level of control come along potentially heavyweight operational

responsibilities that the tenant must take by himself. Slicing support at resource providers can be considered lightweight since they do not need to run large slicing-capable VIM instances;

- **Mode 1 - VIM-dependent slicing:** the slicing in the VIM model, in contrast, is based on multi-tenancy at the VIM level and allows direct inter-domain orchestrator to VIM interaction using a specific allocated shim object. The tenant (Service/Resource) Orchestrator may require changes to use the slice-ready APIs of the Providers' VIM, which may also require deep software changes and customization to support slicing. This VIM-dependent approach requires arguably heavyweight VIMs --becoming a lock-in choice of the Provider-- but frees the tenant from the VIM responsibility;

- **Mode 2 - MANO-based slicing:** the slicing in the Orchestrator model, which uses the inter-domain orchestrator API interaction, a peer to peer approach, where a slice is a set of data structures in the Orchestrator. From the tenant's perspective, depending on the actual split between a Service Orchestrator and a Resource Orchestrator, different software changes and standardized interfaces may be required. Eventually, the tenant BSS/OSS interfaces to the slice-ready APIs of the Provider orchestrator may also need some adaptation;

- **Mode 3 - Service-based slicing:** slicing at the application layer will not provide the relevant slice as a service, and is therefore not considered. This mode of slicing is constrained by the nature of a service-specific vertical solution, requiring per-service Tenant-Provider APIs plus eventual slicing support in the underlying layers, including multi-domain interactions.

As NECOS is building a Slice as a Service platform, and the LSDC (the Lightweight Software Defined Cloud), we considered and evaluated the slicing alternatives with a particular focus on the lightweightness, it being software-defined, and the proposition of allocating slices as a service, again a soft feature, rather than being pre-defined.

Mode 0 is a good fit for NECOS, as we can allocate VIMs as required, all under software control; we can compose slice parts under software control, and it allows for lightweight realizations based on the composition of modular end-to-end stacks fully defined and controlled by the tenant. Slicing at this level requires few, if any, changes to existing components, and can use very lightweight components in many instances, which are easy to allocate at run-time.

Mode 1 is also a good fit for NECOS, as it provides most of the features of mode 0, but replaces the VIM on-demand approach by providing the realization of slicing through extensions to Provider-controlled VIMs. Slicing at this level requires that the VIM knows about slicing, or has a mechanism to pretend that independent slices exist. To maintain a level of lightweightness, it is possible to dynamically create a small shim for each slice under software control.

For both mode 0 and mode 1, we can build an architecture to support Slice as a Service and the LSDC concept, which has very few differences for each mode. For mode 2, the level of slicing is higher up the stack, and it does not provide the level of modularity and simplicity that we see in mode 0 and mode 1. Furthermore, mode 2 requires a different architecture to support slices at this level, and it does not fit with the project Slice as a Service view, as it constraints the levels of freedom by the Tenant. Isolation capabilities plus complex multi-domain issues, plus the requirement to make various changes / additions / removals at the orchestration level. We also observe that other initiatives and projects (e.g., [5GEx], [SONATA], [5G-Transformer], 5G slicing survey [SS1]) have or are looking at this approach to slicing as it has the easiest entry position, but is far more difficult conceptually as well as to actually implement. Finally, Mode 3 is presented here in this document for completeness but is out of scope and a poor fit for NECOS due to its service-specific nature.

Later in this document, in the section on Interactions between the Functional Blocks / Cross Domain Spanning, we show how the modes manifest with respect to the architecture, and present the outcome of a slice spanning across multiple domains.

3 Architectural overview

This section presents an overview of the design of the overall NECOS system architecture required to support Slice as a Service. We have identified all the main architectural functions required for the implementation of NECOS. The architectural design will allow for performing the required service and resource orchestration in a federated and multi-cloud environment way, using the Slice as a Service concept.

Slicing is a move towards segmentation of resources and deployment of virtual elements for the purpose of enhanced services and applications on a shared infrastructure. The model we have for NECOS has the Service Orchestrator interacting with a Slice over the currently existing resources (Data Center and Network resources), as shown in Figure 2.

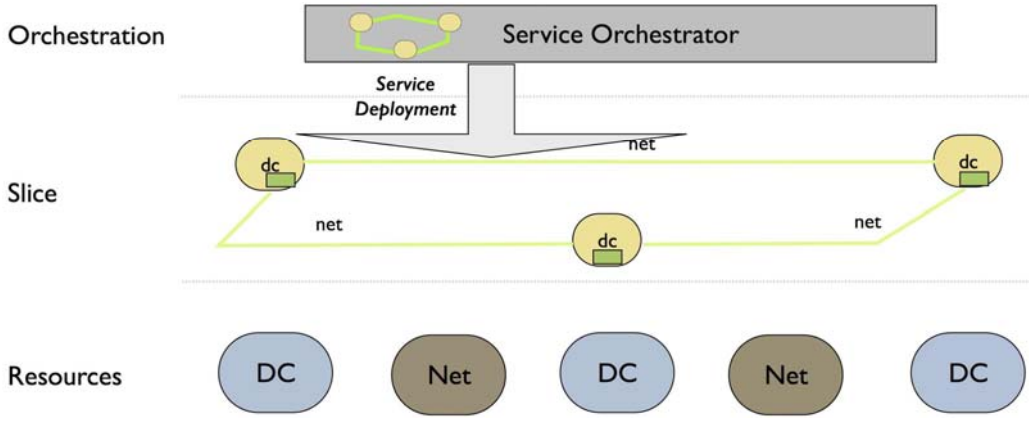

Figure 15. Model of Service Orchestrator interaction with a slice

Currently, the approach is that an Orchestrator performs service deployment across some federated resources, where there are no slices.

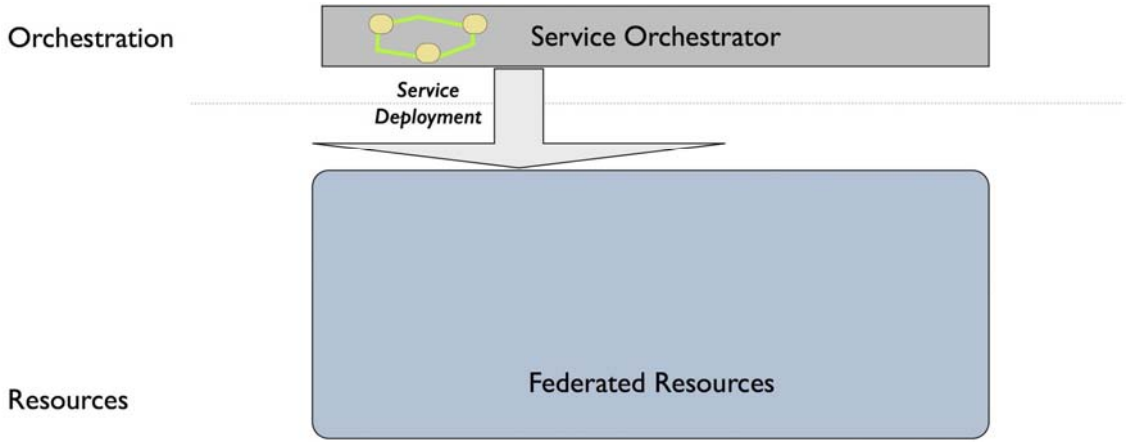

Figure 16. Current Service Orchestrator interaction with federated resources

Differently, for NECOS, we plan to have a Slice as a Service where an Orchestrator performs service deployment across some federated resources, which are aware of the slice concept and can manage their resources accordingly with the slices, thus allowing the approach in Figure 4 that depicts the model we have designed for.

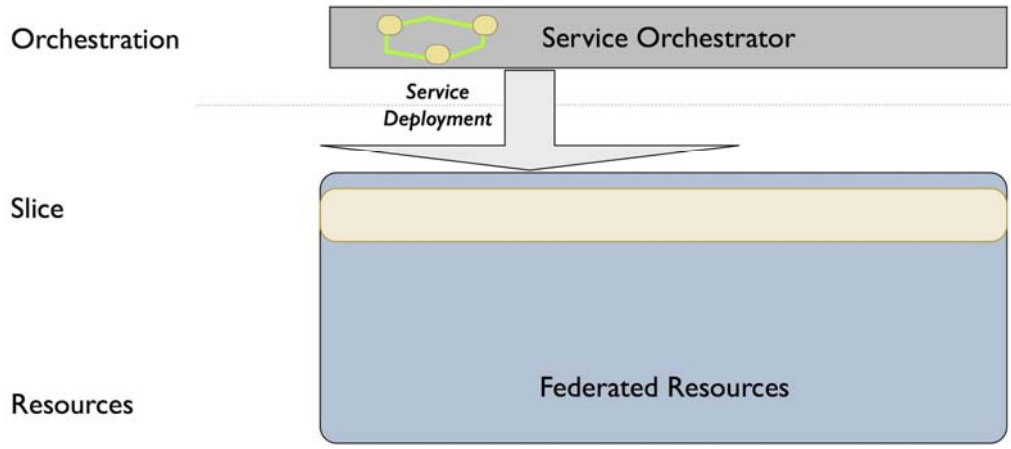

Figure 17. Service Orchestrator and sliced federated resources in NECOS

The idea is that, the Service Orchestrator performs service deployment directly to the slice with as little change as possible to the way in which current service orchestrators are implemented. To achieve this goal, the slice should look like a small instance of the resources.

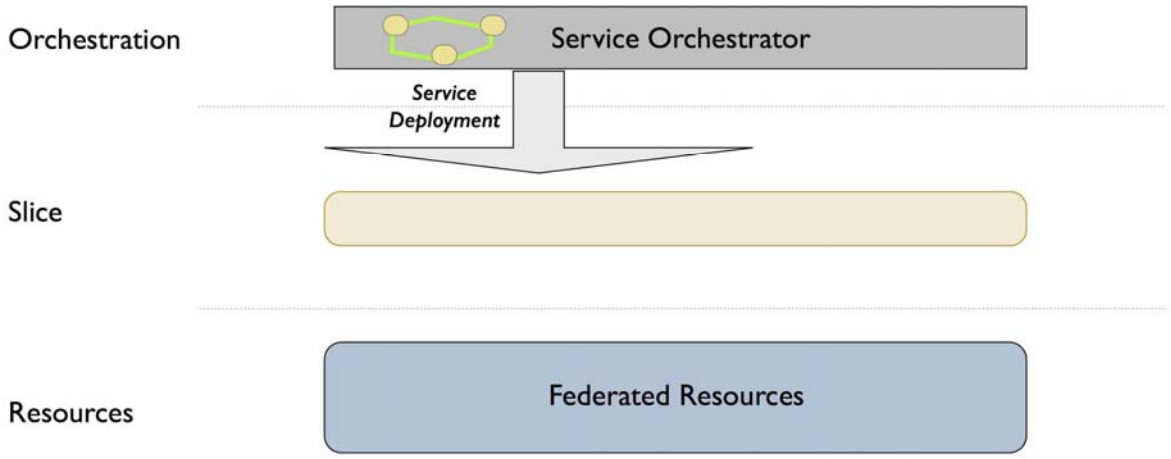

Figure 18. Service Orchestrator interaction with federated resources seen as a slice

3.1 Prioritization of requirements

In this section, we present aspects related to the requirements derived from D2.1 and their impact on the design of the NECOS architecture. In particular, we focus on those requirements that allow us to build a Slice as a Service platform and the LSDC. As NECOS is building a Slice as a Service platform, for the architecture we focus mainly on requirements that are related to providing a slice. We assume aspects related to services are either existing facets, which we will not break, or will be handled by service-related components, which are out of scope, research-wise, for NECOS.

Below is the list of functional and non-functional requirements that have been derived from the analysis of the use-cases in D2.1. They are an aggregate from the perspective of the use-case providers and include requirements for both the service and the underlying infrastructure. As stated, for the purposes of the architecture, we focus on the Slice-based ones. These requirements in green have been taken forward into the architectural design to support the following characteristics: (i) being service-agnostic but being able to adapt the slices to the desired service characteristics; (ii) automating the process of slice configuration in multi-cloud environments; and (iii) providing uniform management.

Table 2. Prioritized requirements (in green background)

5G Infrastructure (vRAN) Scenario, 3 Functional Requirements and 3 Non-Functional Requirements	
RF.vRAN.1	*Service Level Agreement*
RF.vRAN.2	*Accountability*
RF.vRAN.3	*On-demand slice provisioning*
RN.vRAN.1	*Isolation of slice resources*
RN.vRAN.2	*Fairness*
RN.vRAN.3	*Fault detection*
5G Services scenario, 4 Functional Requirements and 3 Non-Functional Requirements	
RF.5G.1	*Service Level Agreement*
RF.5G.2	*Accountability*
RF.5G.3	*On-demand slice provisioning*
RF.5G.4	*External control and management of the offered slices*
RN.5G.1	*Isolation of slice resources*
RN.5G.2	*Fairness*
RN.5G.3	*Fault detection*
vCPE Scenario, 8 Functional Requirements and 5 Non-Functional Requirements	
RF.vCPE.1	*On-demand slice provisioning*
RF.vCPE.2	*Manageable slice*
RF.vCPE.3	*VIM-independence*
RF.vCPE.4	*Bare-metal slice*
RF.vCPE.5	*Lightweight virtualization*
RF.vCPE.6	*Elasticity*
RF.vCPE.7	*Zero touch service provisioning*
RF.vCPE.8	*Fault detection*
RN.vCPE.1	*Isolation of slice resources*
RN.vCPE.2	*SLA monitoring (QoS)*
RN.vCPE.3	*Low latency*
RN.vCPE.4	*High throughput*

RN.vCPE.5	*High availability*
Content Delivery Touristic Services Scenario, 5 Functional Requirements and 5 Non-Functional Requirements	
RF.Touristic(CD).1	*Slice and slice-resource management*
RF.Touristic(CD).2	*Automated Virtual Machine deployment*
RF.Touristic(CD).3	*Traffic load-balancing for content delivery*
RF.Touristic(CD).4	*Slice resource and service monitoring*
RF.Touristic(CD).5	*Service planning*
RN.Touristic(CD).1	*Transparent end-user performance*
RN.Touristic(CD).2	*Heterogeneity handling*
RN.Touristic(CD).3	*Elasticity*
RN.Touristic(CD).4	*Resource-efficiency*
RN.Touristic(CD).5	*Scalability*
Applications Touristic Services Scenario, 3 Functional Requirements and 4 Non-Functional Requirements	
RF.Touristic(APP).1	*Service function chain orchestration*
RF.Touristic(APP).2	*Resource and user-demand prediction capabilities*
RF.Touristic(APP).3	*Resource offloading between edge, core clouds and cloud providers*
RN.Touristic(APP).1	*Resource federation and intelligent multi-domain orchestration*
RN.Touristic(APP).2	*Scalability*
RN.Touristic(APP).3	*Efficient next-generation touristic application performance*
RN.Touristic(APP).4	*Elasticity*
Emergency Scenario, 4 Functional Requirements and 3 Non-Functional Requirements.	
RF.emergency.1	*Dynamic slice management*
RF.emergency.2	*Dynamic service definition*
RF.emergency.3	*Timely slice management*
RF.emergency.4	*Orchestration*
RN.emergency.1	*High Reliability*
RN.emergency.2	*High Availability*

RN.emergency.3	*High Survivability*

As current cloud architectures are lacking the essential features such as the creation of separate cloud slices to be used in isolation by different customers to provide effective service elasticity, the discovery of slices, and mechanisms for cross-domain slice federation, we focus on these requirements to create a suitable architecture.

3.1.1 Synthesis - Analysis and Ranking of the Functional and Non-Functional Requirements

For each scenario in NECOS project (i.e., 5G Networks Scenario, vCPE Scenario, Touristic Services Scenario, Emergency Services Scenario) we evaluated correlations between the identified requirements and NECOS Critical Success Factors/NECOS Key Performance Indicators/NECOS Expected Differentiated Factors (Characteristics). In other words, we evaluated how each requirements is contributing to solve/enable each Project Critical Success Factors/Project Performance Indicators/Project Expected Differentiated Characteristics as seen from each scenario.

The Critical Success Factors/Performance Indicators/Expected Differentiated Characteristics used in the QFD analysis are as follows:

Critical Success Factors (CSF):

- CSF1 - Isolation-isolation is the factors that distinguish slicing from other cloud-based solutions. Since the slices are isolated from each other in all network, computing, and storage planes, the user experience of the slice will be the same as if it was a physically separate infrastructure;
- CSF2 - Cost Reducing Any virtualization solution, being sliced or not, must benefit from its economy of scale, a multi-domain solution such as NECOS has to take advantage of the benefits of the market economy reaching scale through multiple providers, potentially specialized in their own virtualization domain (i.e., virtual networks, virtual computing, etc.) and therefore reducing its own costs and at the end, the costs transferred to the users;
- CSF3 - Reliability The redundancy, geographic distribution, and technology diversity provided by a service that orchestrates different connectivity and computing services from different providers is a key factor for its reliability. As long as the orchestrator has the ability to quickly re-provision a slice after a failure is detected the reliability of the overall slice service may be higher than the reliability offered by each provider;
- CSF4 - Flexibility of cloud-based services should be one of its main distinguishing features. Much more on a multi-domain, diverse platform, as far as it works efficiently and transparently for the user. It is also closely related with reducing the costs for the user, as the resources used by the slice, and therefore its cost, can be reduced or augmented following the demands of the service deployed on top of it, avoiding overprovisioning. Additionally, the tenant must consider the flexibility of a lightweight solution such as NECOS, which intends to offer a basic service inside of which the tenant deploys its own services following its own policies and requirements;
- CSF5 - Scalability - A particular case of such flexibility is the scalability of a provisioned slice. There are several definitions of scalability and elasticity, the CSF described below. For the scope of this project we define scalability as the ability to increase workload size within existing infrastructure (hardware, software, etc.) without impacting performance. We can think in scalability in two different dimensions: scalability of a particular provisioned slice and scalability of the number and size of the slices provided;
- CSF6 - Elasticity As with scalability, there are more than one definition for elasticity. In the context of this project, elasticity is the ability to grow or shrink slice resources dynamically as needed to adapt to workload changes in an autonomic manner, maximizing the use of resources;

- CSF7- Security Being a standard critical feature for any system, the security must be of particular attention for a solution based on sharing resources, isolation, and avoiding side channels must be a priority;
- CSF8 - Slice Efficiency The critical factors mentioned above are no relevant if the communications and computing services provided by a slicing solution do not reach the performance expected and paid by the tenants/users;
- CSF9 - Lifecycle Efficiency Additionally, a slicing service must be efficient regarding the own slice lifecycle: provisioning, monitoring, recovering from failures, etc. This efficiency is closely related with most of the factors mentioned above, from it depend factors such as cost, flexibility, and reliability;
- CSF10 - Simplicity is obviously a factor of importance from the point of view of the client but also an aspect with impact on other CSFs such as cost efficiency, but also on reliability and security. Understanding clearly what is being configured and provisioned is the first step to implement a service inexpensive, reliable and secure.

Key Performance Indicators (KPI):

- KPI1-Average elasticity response time (in seconds);
- KPI2-Average end-to-end delay (in milliseconds, measured as half average RTT);
- KPI3-Average service provisioning time (in seconds);
- KPI4-Average slice provisioning time (in seconds);
- KPI5-Average throughput (in Mbps);
- KPI6-End-to-end slice availability (% of time);
- KPI7-Monitoring-data availability (%);
- KPI8 -Number of application users;
- KPI9-Physical Server utilization;
- KPI10-Service demand prediction accuracy;
- KPI11-Service disruption index;
- KPI12-Service QoE;
- KPI13-SLA fulfilment index;
- KPI14-Average Slice Decommission time (in seconds);
- KPI15-Slice isolation index;
- KPI16-Average Slice Provisioning time (in seconds).

Expected Differentiated Factors (Characteristics) (DF):

- DF1 - LSDC empowers a new service model – the Slice as a Service, by dynamically mapping service components to a slice. The enhanced management for the infrastructure creates slices on demand and slice management takes over the control of all the service components, virtualized network functions and system programmability functions assigned to the slice, and (re) configure them as appropriate to provide the end-to-end service;
- DF2 - LSDC enables easy reconfiguration and adaptation of logical resources in a cloud networking infrastructure, to better accommodate the QoS demand of the Slice, through using software that can describe and manage various aspects that comprise the cloud environment;
- DF3 - LSDC allows each aspect of the cloud environment – from the networking between virtual machines to the SLAs of the hosted applications – to be managed via software. This reduces the complexity related to configuring and operating the infrastructure, which in turn eases the management of the cloud infrastructure;
- DF4 - LSDC platform will offer the ability to a specific cloud provider to federate his own infrastructure with other cloud providers with different configurations in order to realize virtualized services through the use of the Slice as a Service concept. The users of the LSDC APIs and platform will be able to create virtual services that can span the merged cloud infrastructure offered by different cloud providers. This concept is not purely technical, it can also encompass business, cultural, geographical or in any other domain.

Correlation scores and importance factors for each problem were provided by an expert team of 17 members from NECOS partners. These correlation evaluations were done according to each scenario for each Critical Success Factors/ Performance Indicators/ Expected Differentiated Characteristics and also for all scenarios.

The requirements were ranked for importance and impact on Critical Success Factors/ Key Performance Indicators/ Expected Differentiated Characteristics using the Quality Function Deployment (QFD) method, and full results are presented in Appendix 3. The following tables represent the summary of the prioritisation results.

The ordering of the aggregated requirements will be used in the detailed design and implantation of the NECOS functional components.

Requirements contributing most (✓) / least (✗) to realising Critical Success Factors/ Performance Indicators/ Expected Differentiated Characteristics if a Scenario is considered in isolation are depicted in the following table. The rest of the requirements are emerging as having average importance in realising Critical Success Factors/ Performance Indicators/ Expected Differentiated Characteristics as far as each Scenario is concerned.

Table 3. Requirements contributing most / least to realising Critical Success Factors

Index	Requirement ID & Name	Scenario 5G Networks	Scenario vCPE Scenario	Scenario Touristic Scenario	Scenario Emergency Scenario	Scenario All Scenaros
1	*RF.vRAN.1-Service Level Agreement*			✗	✗	
2	*RF.vRAN.2-Accountability*					
3	*RF.vRAN.3-On-demand slice provisioning*	✗	✗	✗	✗	✗
4	*RF.vRAN.34-Isolation of slice provisioning*	✓	✓		✓	✓
5	*RN.vRAN.5 -Fairness*	✓	✓	✓	✓	✓
6	*RN.vRAN.6 -Fault detection*	✗	✗	✗	✗	✗
7	*RF.5G.1-Service Level Agreement*		✗	✗	✗	
8	*RF.5G.2-Accountability*	✗	✗	✗	✗	✗
9	*RF.5G.3-On-demand slice provisioning*	✓	✓	✓	✓	✓
10	*RF.5G.4-External control and management of the offered slices*	✓	✓	✓	✓	✓
11	*RN.5G.5-Isolation of slice resources*	✓	✓	✓	✓	✓
12	*RN.5G.6-Fairness*					
13	*RN.5G.7-Fault detection*					
14	*RF.vCPE.1-On-demand slice provisioning*	✗	✗	✗	✗	✗
15	*RF.vCPE.2-Manageable slice*	✗	✗	✗	✗	
16	*RF.vCPE.3-VIM-independence*		✗	✗		✗
17	*RF.vCPE.4-Bare-metal slice*		✗	✗		
18	*RF.vCPE.5-Lightweight virtualization*		✗	✗		
19	*RF.vCPE.6-Elasticity*					
20	*RF.vCPE.7-Zero touch service provisioning*					
21	*RF.vCPE.8-Fault detection*					
22	*RN.vCPE.9-Isolation of slice resources*					
23	*RN.vCPE.10-SLA monitoring (QoS)*	✓	✓	✓	✓	
24	*RN.vCPE.11-Low latency*					✓
25	*RN.vCPE.12-High throughput*					
26	*RN.vCPE.13-High availability*					
27	*RF.Touristic(CD).1-Slice and slice-resource management*	✓	✓	✓	✓	
28	*RF.Touristic(CD).2-Automated Virtual Machine deployment*	✓	✓	✓	✓	✓
29	*RF.Touristic(CD).3-Traffic load-balancing for content delivery*					✓
30	*RF.Touristic(CD).4-Slice resource and service monitoring*		✓	✓	✓	
31	*RF.Touristic(CD).5-Service planning*					✓
32	*RN.Touristic(CD).6-Transparent end-user performance*					
33	*RN.Touristic(CD).7-Heterogeneity handling*					
34	*RN.Touristic(CD).8-Elasticity*					
35	*RN.Touristic(CD).9-Resource-efficiency*	✓				
36	*RN.Touristic(CD).10-Scalability*					
37	*RF.Touristic(APP).1 Service function chain orchestration*	✓	✓	✓	✓	✓
38	*RF.Touristic(APP).2 Resource and user-demand prediction capabilities*					
39	*RF.Touristic(APP).3 Resource offloading between edge, core clouds and cloud providers*		✓	✓	✓	✓
40	*RF.Touristic(APP).4 Resource federation and intelligent multi-domain orchestration*					
41	*RF.Touristic(APP).5 Scalability*	✓				
42	*RF.Touristic(APP).6 Efficient next-generation touristic application performance*					
43	*RF.Touristic(APP).7 Elasticity*		✓	✓	✓	✓
44	*RF.emergency.1 Dynamic slice management*					
45	*RF.emergency.2 Dynamic service definition*	✓				
46	*RF.emergency.3 Timely slice management*					
47	*RF.emergency.4 Orchestration*	✓		✓	✓	✓
48	*RF.emergency.5 High Reliability*	✓	✓	✓	✓	✓
49	*RF.emergency.6 High Availability*					
50	*RF.emergency.7 High Survivability*	✓	✓	✓	✓	✓

Requirements contributing most (✓) / least (✗) to realising Critical Success Factors if a Scenario is considered in isolation are depicted in the following table. The rest of the requirements are emerging as having average importance in realising project outcomes as far as each Actor is concerned.

Table 4. Requirements contributing most /least to realising Key Performance Indicators

Index	Requirement ID & Name	Scenario 5G Networks	Scenario vCPE Scenario	Scenario Touristic Scenario	Scenario mergency Scenar	Scenario All Scenaros
1	*RF.vRAN.1-Service Level Agreement*	✓		✓	✓	✓
2	*RF.vRAN.2-Accountability*			✓	✓	
3	*RF.vRAN.3-On-demand slice provisioning*		✗			
4	*RF.vRAN.34-Isolation of slice provisioning*		✓			
5	*RN.vRAN.5 -Fairness*		✓			
6	*RN.vRAN.6 -Fault detection*	✗	✗	✗	✗	✗
7	*RF.5G.1-Service Level Agreement*		✗			
8	*RF.5G.2-Accountability*		✗			
9	*RF.5G.3-On-demand slice provisioning*	✗	✓	✗	✗	✗
10	*RF.5G.4-External control and management of the offered slices*	✓	✓	✓	✓	✓
11	*RN.5G.5-Isolation of slice resources*	✓	✓	✓	✓	✓
12	*RN.5G.6-Fairness*					
13	*RN.5G.7-Fault detection*					
14	*RF.vCPE.1-On-demand slice provisioning*		✗			
15	*RF.vCPE.2-Manageable slice*	✗	✗	✗	✗	✗
16	*RF.vCPE.3-VIM-independence*	✗	✗	✗	✗	✗
17	*RF.vCPE.4-Bare-metal slice*	✗	✗	✗	✗	✗
18	*RF.vCPE.5-Lightweight virtualization*		✗			
19	*RF.vCPE.6-Elasticity*	✓		✓	✓	✓
20	*RF.vCPE.7-Zero touch service provisioning*					
21	*RF.vCPE.8-Fault detection*					
22	*RN.vCPE.9-Isolation of slice resources*					
23	*RN.vCPE.10-SLA monitoring (QoS)*		✓		✗	
24	*RN.vCPE.11-Low latency*	✗		✗	✗	✗
25	*RN.vCPE.12-High throughput*	✗		✗	✗	✗
26	*RN.vCPE.13-High availability*				✗	
27	*RF.Touristic(CD).1-Slice and slice-resource management*	✓	✓	✓	✓	✓
28	*RF.Touristic(CD).2-Automated Virtual Machine deployment*		✓			
29	*RF.Touristic(CD).3-Traffic load-balancing for content delivery*					
30	*RF.Touristic(CD).4-Slice resource and service monitoring*		✓		✗	
31	*RF.Touristic(CD).5-Service planning*					
32	*RN.Touristic(CD).6-Transparent end-user performance*					
33	*RN.Touristic(CD).7-Heterogeneity handling*				✗	
34	*RN.Touristic(CD).8-Elasticity*			✓	✓	✓
35	*RN.Touristic(CD).9-Resource-efficiency*	✓				
36	*RN.Touristic(CD).10-Scalability*					
37	*RF.Touristic(APP).1 Service function chain orchestration*		✓			
38	*RF.Touristic(APP).2 Resource and user-demand prediction capabilities*				✗	
39	*RF.Touristic(APP).3 Resource offloading between edge, core clouds and cloud providers*		✓		✗	
40	*RF.Touristic(APP).4 Resource federation and intelligent multi-domain orchestration*				✗	
41	*RF.Touristic(APP).5 Scalability*					
42	*RF.Touristic(APP).6 Efficient next-generation touristic application performance*					
43	*RF.Touristic(APP).7 Elasticity*		✓			
44	*RF.emergency.1 Dynamic slice management*					
45	*RF.emergency.2 Dynamic service definition*				✗	
46	*RF.emergency.3 Timely slice management*	✓		✓	✓	✓
47	*RF.emergency.4 Orchestration*					
48	*RF.emergency.5 High Reliability*		✓			
49	*RF.emergency.6 High Availability*					
50	*RF.emergency.7 High Survivability*		✓		✗	

Table 5. Requirements contributing most / least to realising Expected Differentiated Characteristics

Index	Requirement ID & Name	Scenario 5G Networks	Scenario vCPE Scenario	Scenario Touristic Scenario	Scenario Emergency Scenario	Scenario All Scenaros
1	*RF.vRAN.1-Service Level Agreement*	▯	▯	▯	▯	▯
2	*RF.vRAN.2-Accountability*					
3	*RF.vRAN.3-On-demand slice provisioning*					
4	*RF.vRAN.34-Isolation of slice provisioning*					
5	*RN.vRAN.5 -Fairness*					
6	*RN.vRAN.6 -Fault detection*	▯	▯	▯	▯	▯
7	*RF.5G.1-Service Level Agreement*				▯	
8	*RF.5G.2-Accountability*	▯	▯		▯	
9	*RF.5G.3-On-demand slice provisioning*	▯	▯	▯	▯	▯
10	*RF.5G.4-External control and management of the offered slices*				▯	▯
11	*RN.5G.5-Isolation of slice resources*	▯	▯		▯	▯
12	*RN.5G.6-Fairness*	▯	▯	▯	▯	
13	*RN.5G.7-Fault detection*	▯	▯	▯	▯	
14	*RF.vCPE.1-On-demand slice provisioning*	▯	▯	▯	▯	
15	*RF.vCPE.2-Manageable slice*	▯	▯	▯	▯	▯
16	*RF.vCPE.3-VIM-independence*			▯		▯
17	*RF.vCPE.4-Bare-metal slice*	▯	▯		▯	▯
18	*RF.vCPE.5-Lightweight virtualization*	▯	▯	▯	▯	
19	*RF.vCPE.6-Elasticity*	▯	▯	▯	▯	▯
20	*RF.vCPE.7-Zero touch service provisioning*			▯		
21	*RF.vCPE.8-Fault detection*					
22	*RN.vCPE.9-Isolation of slice resources*					
23	*RN.vCPE.10-SLA monitoring (QoS)*	▯	▯		▯	
24	*RN.vCPE.11-Low latency*	▯	▯	▯	▯	
25	*RN.vCPE.12-High throughput*	▯	▯	▯		▯
26	*RN.vCPE.13-High availability*			▯		▯
27	*RF.Touristic(CD).1-Slice and slice-resource management*					
28	*RF.Touristic(CD).2-Automated Virtual Machine deployment*					▯
29	*RF.Touristic(CD).3-Traffic load-balancing for content delivery*	▯	▯		▯	
30	*RF.Touristic(CD).4-Slice resource and service monitoring*			▯		
31	*RF.Touristic(CD).5-Service planning*					
32	*RN.Touristic(CD).6-Transparent end-user performance*		▯	▯	▯	
33	*RN.Touristic(CD).7-Heterogeneity handling*	▯	▯	▯	▯	
34	*RN.Touristic(CD).8-Elasticity*	▯	▯	▯	▯	▯
35	*RN.Touristic(CD).9-Resource-efficiency*	▯				
36	*RN.Touristic(CD).10-Scalability*					
37	*RF.Touristic(APP).1 Service function chain orchestration*					
38	*RF.Touristic(APP).2 Resource and user-demand prediction capabilities*		▯	▯	▯	
39	*RF.Touristic(APP).3 Resource offloading between edge, core clouds and cloud providers*					
40	*RF.Touristic(APP).4 Resource federation and intelligent multi-domain orchestration*					
41	*RF.Touristic(APP).5 Scalability*	▯	▯	▯	▯	
42	*RF.Touristic(APP).6 Efficient next-generation touristic application performance*	▯	▯	▯	▯	
43	*RF.Touristic(APP).7 Elasticity*					
44	*RF.emergency.1 Dynamic slice management*	▯	▯	▯	▯	
45	*RF.emergency.2 Dynamic service definition*	▯	▯	▯	▯	
46	*RF.emergency.3 Timely slice management*					▯
47	*RF.emergency.4 Orchestration*					
48	*RF.emergency.5 High Reliability*					
49	*RF.emergency.6 High Availability*	▯	▯	▯	▯	
50	*RF.emergency.7 High Survivability*	▯	▯	▯	▯	

The importance of requirements for realising Critical Success Factors based on combined all scenarios viewpoints is presented in Figure 6.

Figure 19. Importance of requirements for realising Critical Success Factors based on combined all scenarios

The importance of requirements for realising Key Performance Indicators based on combined all scenarios viewpoints is presented Figure 7.

Figure 20. Importance of requirements for realising Key Performance Indicators based on combined all scenarios

The importance of requirements for realising Expected Differentiated Characteristics based on combined all scenarios viewpoints is presented in the following Figure 8.

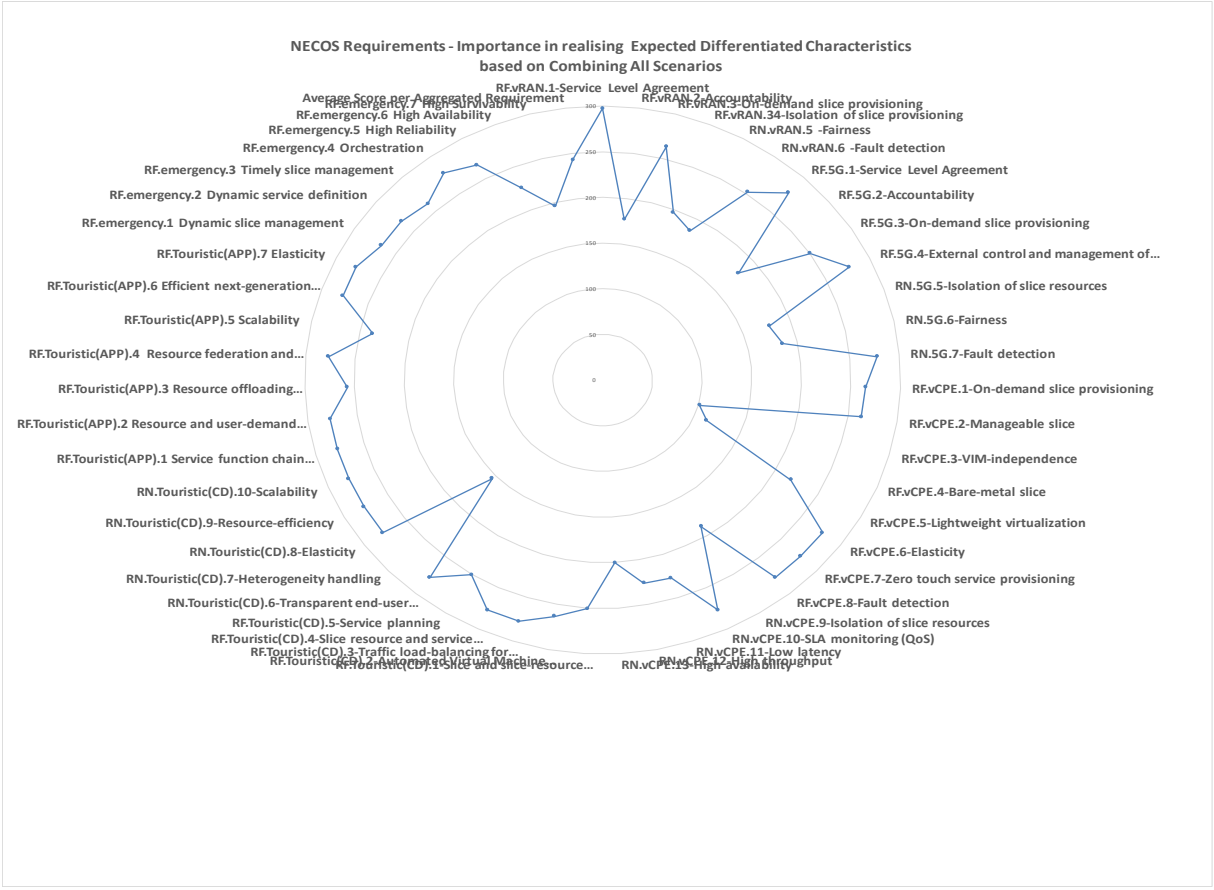

Figure 21. Importance of requirements for realising Expected Differentiated Characteristics based on combined all scenarios

3.2 Functional architecture description

As mentioned in Section 2.1, there are various schemes for creating slices across infrastructure domains that are currently being devised. They have different ideas regarding what a slice is, how to envision a slice within the existing systems, and how to manage the lifecycle of a slice, and how it is presented to the requester.

In NECOS, we have gone for a specific slicing model that has as its foundation a mechanism to partition the underlying infrastructure into constituent *slice parts* and then combine these *slice parts* into a full end-to-end slice.

We are particularly focussed on a set of layered abstractions using slicing elements, ensuring that they all fit together for the purpose of service provisioning. The combination of the right abstractions, the right layering, and a separation of concerns in the architectural design will allow NECOS to create a powerful and flexible Slice as a Service mechanism.

In particular, the Slice as a Service approach provides an adaptable control plane, having features for creating, growing, shrinking, and closing slices, as well as adapting slices at the run-time, while considering service requirements and current cloud resource conditions.

In this section, we present the functional architecture that has been devised to support the Slice as a Service approach. The architecture contains three main high-level sub-systems. These are (1) the NECOS (LSDC) Slice Provider (coloured in blue), (2) the Resource Marketplace (coloured in yellow), and (3) the Resource Providers (coloured in green). These sub-systems are provided in order to support

the tenants of NECOS who wish to use Slice as a Service. The tenant of NECOS is expected to be an organisation that requires slices for running their own services. Figure 9 presents these main elements, how they are grouped and the way they interact with the tenants (coloured in red) who use NECOS.

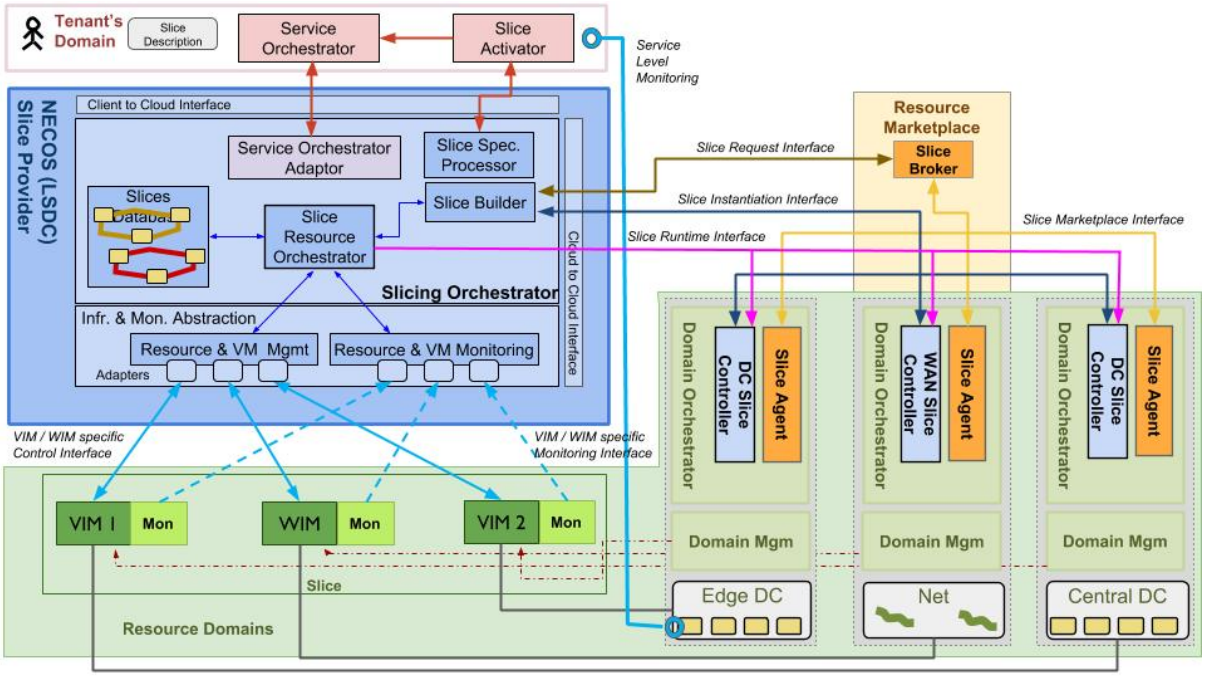

Figure 22. NECOS functional architecture

The following sections will present the details of the NECOS functional architecture.

3.2.1 The NECOS Tenant

The tenant of NECOS is expected to be an organisation that requires slices for running their own services. An example of such a tenant could be a CDN company. The tenant is not expected to be a small organisation that needs a handful of Virtual Machines nor an end-user who wishes to use an online service. Both of these kinds of user already have multiple options available to them.

The NECOS tenant is depicted in the top-left part of Figure 9 – it can be an organisation that is expected to request to a NECOS Slice Provider the instantiation of heterogeneous resources including computation, storage and network in the form of end-to-end slice, and to run their own Service Orchestrator for deploying and managing their own services on that allocated slice. The tenant is likely to have a range of options for choosing a Slice Provider, and such could be based on factors such as commerce, geography, and contractual obligations.

To participate in Slice as a Service environments, and to initiate and interact with slices, the tenant also needs to run their own Slice Activator component., a new component responsible for two main tasks: (i) for allowing a Slice Description to be sent to the NECOS platform; and (ii) for handling the response back from the NECOS Slice Provider. On receiving the response, the NECOS Slice Provider will inform the corresponding tenant that the slice is live and ready, and therefore, the tenant can progress and deploy services onto the slice.

The tenant can also deploy their own monitoring system to check the performance of the services (e.g., service specific KPIs) running on its slice (via the Service Level Monitoring Interface), or it can delegate that to the NECOS provider according to the desired level of abstraction. The monitoring information

related to the service instances will be provided as feedback to the Service Orchestrator so that it can perform proper services' lifecycle management and propagate lifecycle events to the NECOS slicing orchestrator if needed (via the Service Orchestrator Adaptor).

3.2.2 The NECOS Slice Provider (LSDC)

The NECOS Slice Provider (LSDC) is the sub-system that allows for the creation of full end-to-end Slices from a set of constituent Slice Parts. In NECOS, a Slice looks the same as the full set of federated resources, with the main attribute being that the domains look a lot smaller, as they have been sliced. This is why we call the Slice, the LSDC - the Lightweight Software Defined Cloud.

The NECOS Slice Provider (LSDC) presents a northbound API that is compatible with a tenant's Service Orchestrator, thus enabling tenants either to operate on the full infrastructure or to choose to interact with Slice as a Service providers, using NECOS.

When requesting a Slice from a NECOS provider, there is a Slice Builder component that goes out to a specially designed and configured Resource Marketplace that, based on a Slice Specification, is able to find Slice Parts across various participating Resource Domains.

The Slice Builder, with support from the Slice Resource Orchestrator, is the component inside the NECOS Slice Provider (LSDC) in charge of combining the Slice Parts that make up a slice into a single aggregated Slice. The Slice Resource Orchestrator is responsible for the orchestration of the Slices, including their management at the run-time of their lifecycle. It is also responsible to orchestrate the service elements across the Slice parts that make up the full end-to-end slice. Finally, it is the component that is responsible for the actual placement and embedding of VMs and virtual links for the services into the resource domains.

In order to interact with the actual remote cloud elements, the NECOS Slice Provider (LSDC) has an Infrastructure and Monitoring Abstraction (IMA) mechanism. The IMA allows the Slice Provider to interact with various remote VIMs, WIMs, and monitoring sub-systems in a generic way, using plug-in adaptors with the relevant API interactions. The IMA affords the Slice Resource Orchestrator to interact with the remote clouds in order to provision the actual tenant services, and to monitor the remote resources running those services (via additional monitoring data that is not available via the Service Level Monitoring Interface).

3.2.3 The Resource Marketplace

The Resource Marketplace provides the way for the NECOS (LSDC) Slice Provider to find the slice parts to build up a slice. Rather than having a pre-determined set of providers that have been configured in a federation, we chose to use a more flexible model of a marketplace from which we could provision slice parts. This was done for two main reasons: (i) in order to build end-to-end slices we need a mechanism to reach and interact with providers in multiple geographic places - including mobile edge and sensor networks - which are often not covered with existing federation approaches; and (ii) slice creation is generally more dynamic than federation agreements, so we need a highly dynamic run-time mechanism for finding providers. We observed that there are online marketplaces for flights and for hotel rooms that provide a good operational model to follow. There is a broker, which can dynamically go to multiple agents and get a collection of offers for the required resource. The end-user can then make a choice as to which offer is best suited. The agents can be a division of the resource provider, or they may be an external aggregator that provides compound offers. From the perspective of the broker and the end-user, the agent just provides offers at a price point with certain conditions. Such an approach fitted with the NECOS Slice as a Service system very well.

Within the NECOS (LSDC) Slice Provider, the *Slice Builder* is responsible to build a full end-to-end multi-domain slice from the relevant constituent slice parts. In order to do so, the *Slice Builder* has to

find available resources from the marketplace. This search involves communication with a *Slice Broker*, the entity responsible for contacting provider *Slice Agents*. In the NECOS approach, there is expected to be multiple *Slice Brokers* in various locations - maybe multiple per country. Each *Slice Broker* will have access to many *Slice Agents* across many geographical domains, which are able to provide offers for the required slice parts that match a set of request constraints. For example, the slice may need DC slice parts in Spain, Greece, the UK, and in Brazil, with network slice parts that connect the DC slice parts.

Once the various offers for the parts have been collected by the *Slice Broker*, it responds to the *Slice Builder* request with alternative options (i.e., the offers) for each slice part. Then the *Slice Builder* can select the relevant resources for each slice part, based on the original slice request.

To aid in flexibility, we consider that different Marketplaces can be created, setup and configured, based on the use-cases of the various participants. Examples of Marketplaces include:

- *the telecoms Marketplace*, which is a limited setup between close co-operating telecoms providers, i.e., allowing the creation of slices across each other's infrastructure,
- *the contract Marketplace*, which is a setup between partners with a level of trust based on signed contracts, i.e., allowing their resources to be used by others for slices,
- *the open Marketplace*, which allows any provider to offer resources and register their offerings for users.

Other setups can be devised.

3.2.4 The Resource Providers

The *Resource Providers* are those organisations that can provide the resources required for the slice parts - namely, Data Center resources in the form of servers, storage, and Network resources. Further resources can be provided by organisations that have Mobile Edge, Sensor Networks, Wireless, etc. Each resource provider will be capable of providing slice parts, which will be part of a full end-to-end slice. The *Resource Provider* also needs to provision the relevant manager for each slice part, meaning a VIM for a DC slice part, or a WIM for a network slice part. These managers allow those resources to be managed and utilised by the *Slice Resource Orchestrator*.

In order to afford service provisioning over the slices, it is necessary to have a mechanism to support the slicing of the DC compute and storage resources, as well as the network resources. To manifest this slice approach, in NECOS two components at the infrastructure level act as a point of control and management of DC and Network slices. For each data center, a *DC Slice Controller* is in charge of creating DC slices within the data center, allocating the required compute and storage resources for a given Slice part, and returning a handle to a VIM running on it. The VIM can either be a separate instance deployed on demand based on the tenant specification (e.g., if an explicit reference was part of the Slice request) or an existing VIM running in that *Resource Domain* on which an isolated interaction point (such as a shim) was created for the tenant. Similarly, for each network domain, a *WAN Slice Controller* is in charge of instantiating a network slice between two DC slices and either deploying an on-demand WIM or adding an isolated interaction point for the tenant to an existing WIM instance.

The mechanism to find these slice parts was described above, in the *Resource Marketplace* section (3.2.3). To participate in a Marketplace, a resource provider needs to run a *Slice Agent*, which can find available resources within the local resource domain, and then provide an *offer* for those resources to the *Resource Broker* (in the Marketplace), with a specified timespan and pricing model associated to it.

4 Mapping of the functional components to roles

We see that there are fundamental roles that an actor could take, as well as some combined roles, based on the main sub-systems and components that are in the NECOS system. As shown in Figure 10, we see how each of the main domains, featuring the Tenant's Domain (red, on the right), the Resource Providers (green, on the left), the Resource Marketplace Domain (yellow, at the top), and the Slice Provider Domain (blue, in the middle).

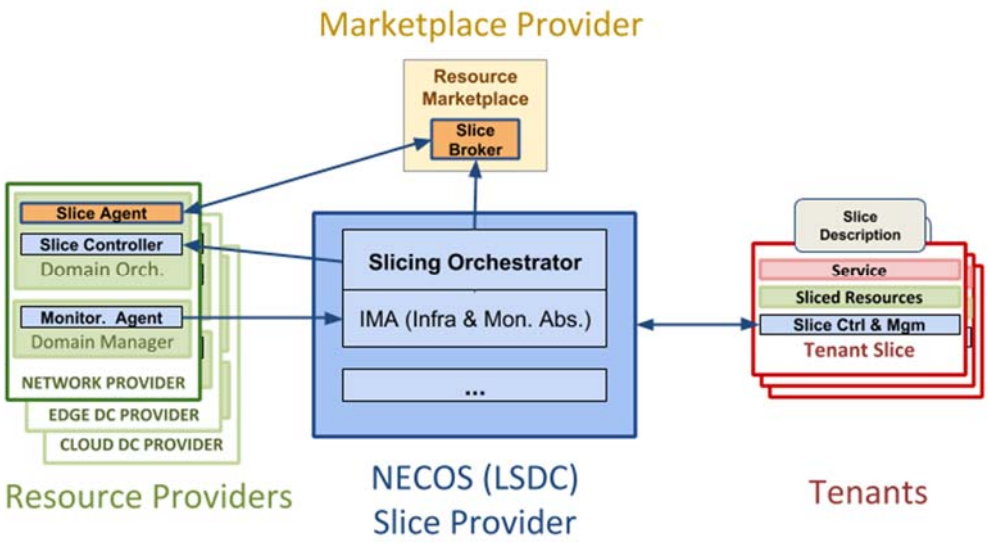

Figure 23. High-level NECOS architecture - per domain

4.1 NECOS fundamental roles

The NECOS functional architecture allows for the identification of different roles that entities can play in the NECOS ecosystem, based in different domains. This role-based view, allows for flexible federation schemes among providers, as it will be argued latter in this section. More specifically, we have identified the following four fundamental roles:

- NECOS (LSDC) Slice Provider Role.
- Data Center Provider Role.
- Network Provider Role.
- Marketplace Broker Role.

Fundamental roles can be assumed by participating to the NECOS ecosystem entities, resulting to entities that might assume more than one roles (i.e., combining roles), as described later in the section.

4.1.1 NECOS Slice Provider (LSDC) role

The **NECOS Slice Provider role** is undertaken by an entity that wishes to offer "slice as a service" to its customers, i.e., the slice tenants. It acts as the main contact point of potential tenants to the NECOS ecosystem. It should be noted that the role does not necessarily assume that the entity has indeed resources to offer, i.e., is a data center or a network provider, since the latter can be offered by other entities in the ecosystem. However, it is considered necessary that architectural components that concern user interaction, such as the Service Orchestrator adaptor are presented. The Role will be responsible for slice creation, thus requires the Slice Specification Processor and Slice Builder components, and the runtime operations (including slice decommissioning), thus Slice Resource Orchestrator and Infrastructure & Monitoring Abstraction components must be present. Figure 11 depicts all the

necessary functional components for the specific role; their detailed descriptions can be found in the corresponding sections of the current document.

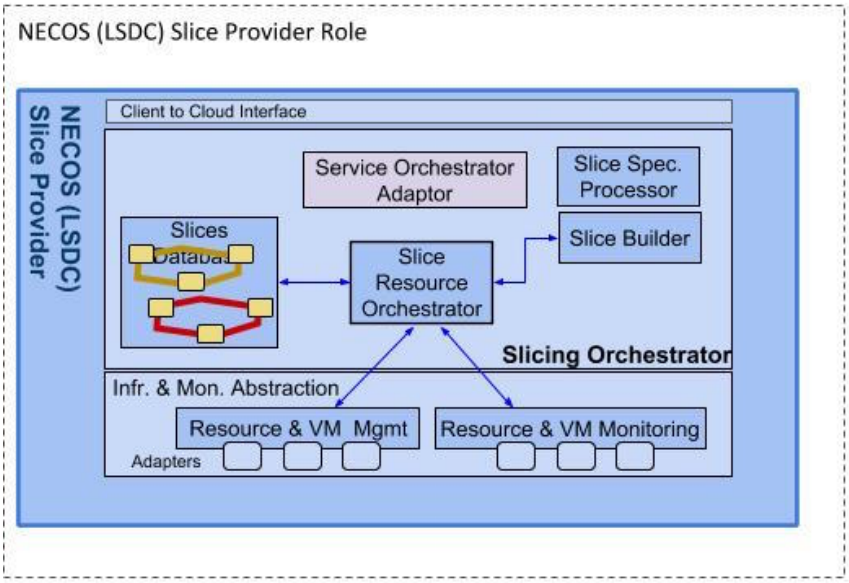

Figure 24. The NECOS Slice Provider role

4.1.2 Data Center and Network Provider roles

Any entity wishing to participating in a NECOS ecosystem simply by offering data center/network resources to be used in slice creation, can undertake either the Data Center or the Network Provider roles, depending on the offered resources.

A **Data Center (DC) provider** (Figure 12 (a)) is assumed to be offering computational resources in either edge and/or core clouds. In order to participate in NECOS ecosystem a DC must have a *Slice Agent* to communicate with a *Broker*, thus participating in at least one *Marketplace*, and a DC Slice controller to handle resource instantiations for a slice, originating from the *Slice Builder*.

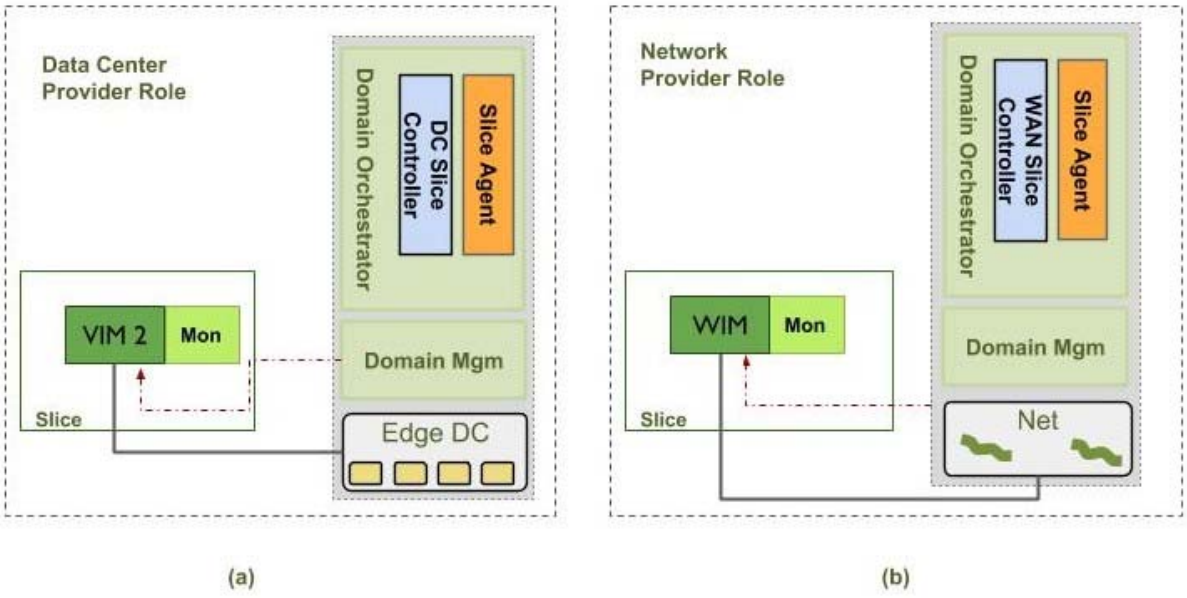

Figure 25. The Data Center (a) and Network Provider (b) roles

A quite similar situation occurs for the NECOS **Network Provider role** (Figure 12 (b)), with the main difference being the type of resources offered. In this case, the *WAN Slice Controller* component must exist in the provider.

4.1.3 Marketplace Broker role

One of the main characteristics of NECOS is the introduction of a resource marketplace, where Brokers act as intermediates between Slice Builders and providers. We consider each marketplace in the NECOS ecosystem to consist of a set of Slice Agents and a single Slice Broker (Figure 13). In this setting separating the Marketplace Broker Role allows any entity to act as an intermediate, and allows more flexibility in slice creation and accommodation of different types of federations.

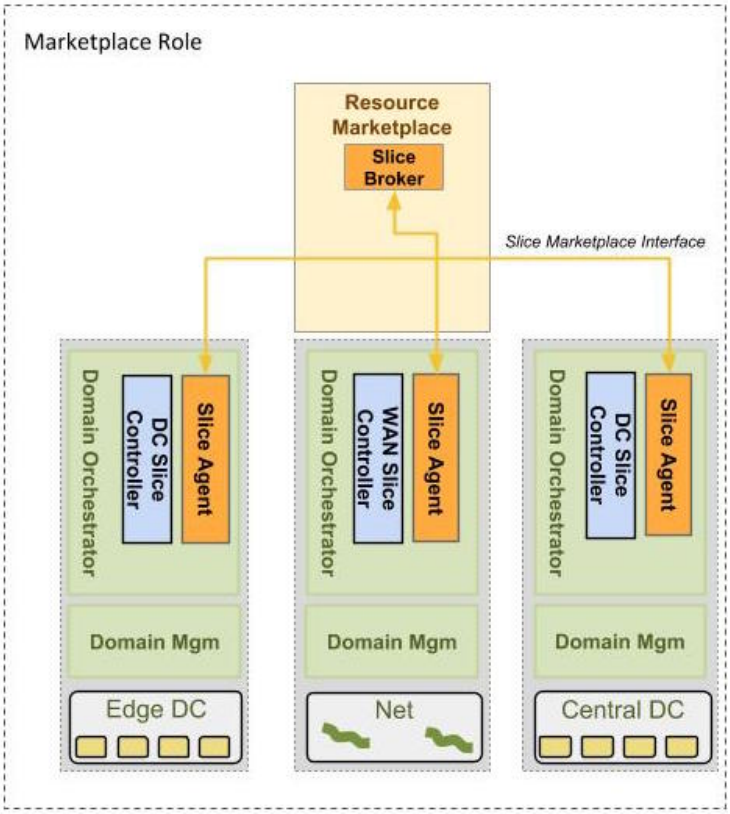

Figure 26. The Marketplace role

4.2 Combining fundamental roles

Obviously, an entity in NECOS can assume more than one fundamental role as described above, following the classic "actors" approach in software engineering.

4.2.1 Combining Data Center and Network Provider roles

For instance, an entity can assume both the roles of the Data Center and Network provider, as shown in Figure 14, offering both compute and network resources.

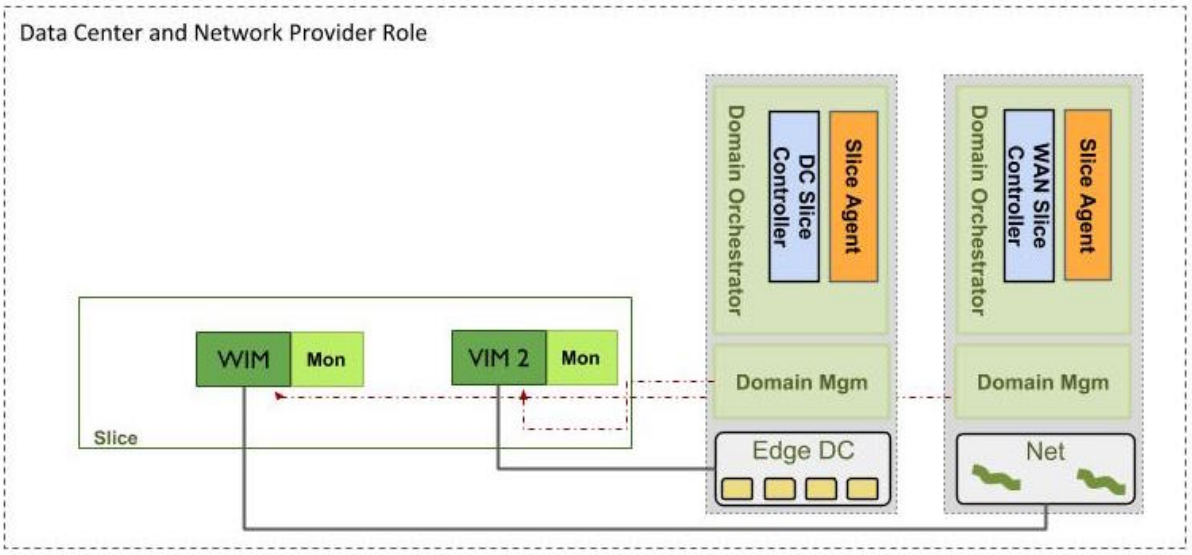

Figure 27. An entity assuming the Data Center and Network Provider role

Similarly, larger entities can be NECOS Slice providers, DC and/or Network providers at the same time (Figure 15), seeking to combine their own resources with resources other entities offer in the NECOS ecosystem. Although not depicted in a corresponding figure, an administrative entity can host Slice Broker architectural components as well.

4.2.2 NECOS Slice Provider and Data Center / Network Provider role

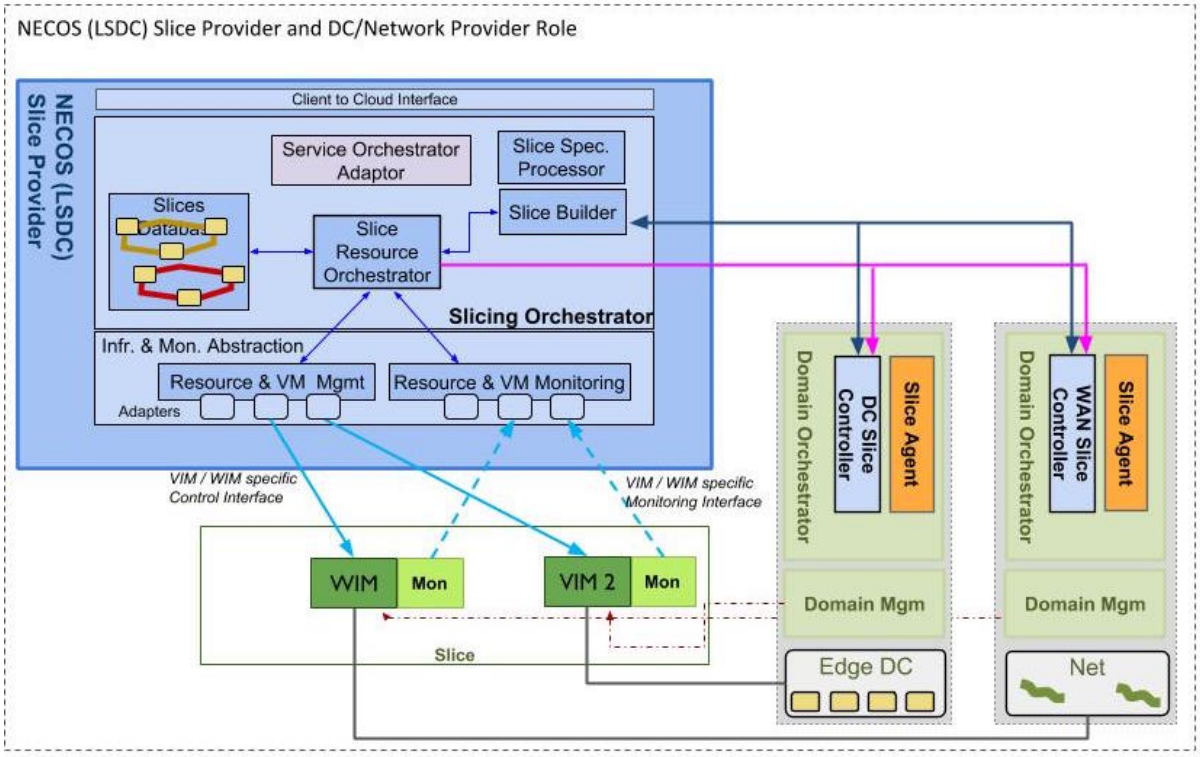

Figure 28. The NECOS Slice Provider and Data Center / Network Provider role

There may be other combinations of roles, but we do not consider these here.

5 Architectural functional blocks

This section describes the main architectural blocks which are in the three main high-level sub-systems described in the previous section. The NECOS (LSDC) Slice Provider, which is the sub-system that allows for the creation of full end-to-end Slices from a set of constituent Slice Parts, has two main functional blocks: (i) the Slicing Orchestrator, and (ii) The Infrastructure & Monitoring Abstraction (IMA) based upon the following components:

Slicing Orchestrator

- Slice Resource Orchestrator
- Service Orchestrator Adaptor
- Slice Specification Processor
- Slice Builder
- Slice Database

Infrastructure & Monitoring Abstraction (IMA)

- Resource and VM Monitoring
- Resource and VM Management
- Adaptors for VIM/WIM Control

There are also the *Resource Domains*, which contain these components:

- DC Slice Controller
- WAN Slice Controller

and the *Marketplace* that has these components:

- Slice Broker
- Slice Agent

All of these functional blocks and their associated components are described in more detail below.

5.1 Slicing Orchestrator

The *Slicing Orchestrator* performs operations to deal with slices via its two main components: (i) the *Slice Builder*, responsible for requesting to the Resource Marketplace the different Slice parts that will be included in an end-to-end Slice via performing an initial orchestration phase to work out a subset of domain that could be used as candidates for the end-to-end slice creation; (ii) the Slice Resource Orchestrator (SRO), which combines the allocated Slice Parts into a single aggregated Slice on which it holds the overall end-to-end view and manages the lifecycle of each individual slice part. Furthermore, the SRO is also responsible for orchestrating the service elements across the Slice parts that make up the full end-to-end slice as it performs the actual placement and embedding of VMs into the resource domains. It also takes care of the lifecycle management of the services running on the slices.

The Slicing Orchestrator takes service lifecycle orchestration decisions based on the information retrieved by the SRO via the Infrastructure Monitoring Abstraction (IMA). This information includes resource facing KPIs corresponding to the service elements that are relevant for the SLA. Lifecycle actions can be triggered according to the collected measurements and the type of service in order to trigger corrective actions on the initial service deployment.

Service lifecycle events can indirectly impact the lifecycle of the slice where they are running. The SRO may receive requests for slice lifecycle events to be performed as consequence of that (this may also have impact on the initial Service Level Agreement stipulated for a slice)

To do all the above-mentioned operations, the Slicing Orchestrator uses the named interfaces and APIs being called:

- *The Slice Request Interface:* provides the mechanisms to the Slice Builder to initiate the instantiation of an end-to-end Slice via interacting with the Slice Broker in the Resource Marketplace. It is used by the Slice Broker to provide the description of the Slice to be allocated, by using *the Slice Marketplace Interface* for interaction between the Slice Broker and the Slice Agents to implement mechanisms for the propagation of resource offerings between (external) resource domains. Different Slice Agents will register their resources availability to the Slice Broker.
- *The Slice Instantiation Interface:* is used by the Slice Builder after available resources offerings have been identified via the Slice Broker; individual resource allocation requests are sent to the relevant Slice Resource Controllers which allocate resources for each single Part of the end to end Slice.
- *The Slice Runtime Interface:* implements the interface that provides functionalities to dynamically modify the resource allocation for a Slice Part. This is required by the Slice Resource Orchestrator to perform lifecycle operation on the end to end Slice according to the feedback received for each Slice Part via the collected monitoring measurements.

Section 6 will provide additional information on how these interfaces are used throughout the lifecycle of the Slice. Further details and the full specification of the API methods are reported in D4.1.

5.1.1 Slice Resource Orchestrator

The Slice Resource Orchestrator (SRO) is responsible for combining the Slice Parts that make up a slice into a single aggregated Slice. It is also responsible for orchestrating the service elements across the Slice parts that make up the full end-to-end slice, and as such it is the component that handles the actual placement and embedding of VMs into the resource domains.

The Slice Resource Orchestrator is not involved in the request for a new slice, but after an end-to-end Slice has been requested by a tenant, via the interaction of the Slice Builder with the different Resource Slice Controllers and the resources for it have been allocated, the NECOS Slice Resource Orchestrator will receive a data model from the Slice Builder related to the new slice. This data model contains: (i) a list of entry points associated to each VIM / WIM that was allocated within a Slice Part, (ii) a list of monitoring end-points within each Slice Part, and (iii) a list of entry points related to the DC Slice Controllers and WAN Slice Controllers that are involved in the Slice allocation process.

5.1.1.1 End-to-End Slice Activation and Registration

The Slice Resource Orchestrator gathers data on the overall view of the slice in terms of allocated computing resources and network topologies, and stores this in the Slice Database, which is the component holding such information. The SRO will be the component interacting with the DC / WAN Slice Controllers (returned by the Slice Builder) to complete the creation of the end-to-end slice. This will be performed by interconnecting each VIM / WIM instance to its closest domain network edge points and then setting up the inter-domain network connectivity between the different Slice Parts.

5.1.1.2 End-to-End Slice Lifecycle

The Slice Resource Orchestrator will take care of performing the slice lifecycle management, i.e., continuously checking that the allocated slice satisfies the requirements that were initially requested by the tenant. In order to do this, the SRO should have access to a set of monitoring measurements (coming from each Slice Part) that provide information about the slice resource utilisation (e.g., the number of available cores, memory, network delay, etc.). This information will be at the granularity of the slice and will not directly be linked to any of the service instances running on that slice. The monitoring

information is expected to be propagated to the SRO via the underlying monitoring abstraction implemented by the IMA. We foresee multiple elasticity scenario where the SRO might be involved in. Later in this document there is a description of some of those scenarios that will be addressed by NECOS.

For instance, if resources in a specific DC Provider need to scale up, the SRO should contact the related resource provider to request the allocation of additional resources accordingly. This will dynamically be carried out by the SRO via the Cloud to Cloud Slice Runtime Interface interacting with the relevant Slice Controllers, in order to re-negotiate the resource allocation in specific Slice Parts (e.g., scaling out the number of physical hosts allocated to a VIM, etc.).

When a slice has to be augmented with resources regardless of their location (or other requirements that cannot be fulfilled with the already allocated Slice Parts), the SRO will contact the Slice Builder to delegate the process of looking for additional resources that can be attached to the existing Slice as new Slice Parts. The Slice Builder will, in turn, contact the Slice Broker in the marketplace so as to discover resources in a similar manner as described for the Slice instantiation phase. Once the handles to the new Slice Parts have been successfully returned back from the Slice Builder, the SRO will perform the operations required to attach those Parts to the existing Slice instance and will update the Slice topology in the Slice Database.

5.1.1.3 Decommissioning End-to-End Slice

In this case, the Slice Resource Orchestrator contacts the list of all DC/WAN Slice controllers obtained with the assistance of the Slice database, again via the Slice Runtime Interface, informing them appropriately to release the indicated resources.

5.1.1.4 Instantiation of virtual resources for the services

The Slice Resource Orchestrator will perform the allocation of the resource elements that are required for the instantiation of the service instances (possibly described by Forwarding Graphs) on the global resource view available in each end-to-end slice. This will be performed by deploying service elements (e.g., a Container, a VM, a VNF or a virtual link), on the slice resource view. The orchestrated instantiation of the different service virtual elements will happen via the northbound interface of the Infrastructure & Management Abstraction (IMA). The IMA will take care of adapting the above instantiation requests to the specific underlying VIM/WIM technology. The SRO will have to request to the IMA the allocation of different adapters required to interact with the different Infrastructure Managers running in the Slice Parts. This will happen via the IMA API providing the entry point to a VIM / WIM the adapter should interact with.

5.1.2 Service orchestrator Adaptor

The Service Orchestrator Adaptor is the component that interacts with the tenant's Service Orchestrator. Its northbound interface talks with the southbound interface of the tenant's Service Orchestrator. It provides the features to adapt the calls from the tenant's Service Orchestrator into calls internal to the Slicing Orchestrator. The Service Orchestrator Adaptor will interact with the other components in the Slicing Orchestrator, for various tasks. This Adaptor provides the linkage that allows a created Slice to look like a small version of the whole infrastructure.

Due to the modular nature of the architecture, we expect that there can be different implementations of the Service Orchestrator Adaptor which can be devised to interact with different Service Orchestrators. A new Service Orchestrator Adaptor can be allocated to each tenant, instantiating the relevant one, based on the tenant's request.

5.1.3 Slice Specification Processor

The Slice Specification Processor plays a fundamental role in the interaction between a Tenant and the NECOS system via the Client-to-Cloud API. It takes as input a Slice Specification from the Slice Activator in the Tenant's Domain, and performs the processing tasks required to build a Slice creation request for the Slice Builder. According to the particular API call invoked by the Tenant (i.e., the *create_slice* call discussed in D4.1, Section 5.1), the Slice Spec Processor will be able to handle a variety of tenant's slice requests. These requests can have different levels of detail, possibly specifying:

1. a fully-described low-level description of a slice, or
2. a slice requirements specification, or
3. a service specification.

The above different ways of requesting slices to the NECOS system will imply the usage of different Client-to-Cloud API calls, and will allow a tenant to have the desired level of abstraction while interacting with the system. For example, when using the approach (1), a tenant will be able to provide details on the type of VIM that should be allocated on a given Slice Part, specifying geographical constraints on the Slice resource to be used as access points for the Slice, etc.

On the other hand, the higher the level of abstraction requested by the tenant, the higher the complexity in the translation of the Slice specification into a proper Slice descriptor that can be provided as input to the Slice Builder.

In case (1), the Slice Spec Processor will probably need to check that the input provided by the tenant is syntactically correct according to the expected data model used to describe the Slice. In case (2) and (3), instead, the Slice Spec Processor will have to perform the required processing tasks on the provided input. This will be necessary either to translate some high-level requirements for the Slice (such as delay, computing performance, etc.) or even more abstract constraints coming from the definition of service that will be instantiated on the Slice once created.

5.1.4 Slice Builder

The *Slice Builder* is responsible for building a full end-to-end multi-domain slice from the relevant constituent slice parts. In order to do so, the *Slice Builder* has to search for available resources in the *Resource Marketplace*. This search involves communication with the *Slice Broker*, the entity responsible for contacting provider *Slice Agents*, as described in Section 3.2.3 of this document. Slice part requests must reflect the structure of the slice that the tenant wishes to create, as they are described in deliverable D4.1. *Slice Agents* in the remote domains will respond to the *Slice Broker* in two possible ways: NO, meaning that there is no slice part available to match the slice request, or: YES, standing that here is an *offer* for a slice part that can be used, with a set of timescales and pricing information. The *Slice Broker* collects all of the responses from the *Slice Agents*.

The *Slice Broker* composes a request message enclosing a set of alternative options (i.e., the offers for the resources) for each slice part, and then replies it to the *Slice Builder*. On receiving the response to its request, the *Slice Builder* selects the suitable resources for each slice part by matching the offers and their attributes against the requirement indications of the corresponding slice request. In this way, the *Slice Builder* can select the best options for all the offers available.

On finishing the slice parts selection, the *Slice Builder* proceeds to prepare the slice, which is done by contacting the corresponding *DC Slice Controllers* and/or *WAN Slice Controllers*. This communication leads to the remote domain the task to instantiate the slice part by creating the relevant VIM and/or WIM, as well as associating the monitoring end-point. The addresses for the remote VIM and WIM objects are passed back to the *Slice Builder* in order to fully instantiate the slice at the *Slice Orchestrator*. Eventually, all the slice details that include the handlers for the relevant created VIMs and WIMs are stored in the *Slice Database* to keep track of the deployed slices.

The *Slice Builder* is also responsible to handle any requests coming from the *Slice Resource Orchestrator* for discovering appropriate resources to augment slices at the runtime. In this situation, since these requests concern allocation of resources anywhere in the NECOS ecosystem, the request message to the *Slice Broker* has to reflect the necessary information regarding the current slice topology and provide a set of possible alternative points in the latter, which should be augmented. The process followed in this case is similar to the initial discovery phase described above.

The *Slice Builder*, along with the assistance of the *Slice Broker*, selects the *Slice Controllers* fitting the above request through propagating a call via the *Slice Request Interface*; it will then use the *Slice Instantiation Interface* to request to the *Slice Controllers* the allocation of resources for the new Slice Parts. At the end of the process, the *Slice Builder* will provide, as a response to the SRO, the entry points of the Infrastructure Manager running in the newly allocated Slice Parts in addition to the entry points of the relevant *Slice Controllers*. The SRO will attach the new Slice Parts to the existing Slice, and will update the corresponding infrastructure view on the *Slice Database*.

5.1.5 Slice Database

The Slice Database aims to afford full knowledge about existing Slices in the NECOS ecosystem, which is achieved by storing all the information related to the topology of each slice. It keeps information on the slice parts, which can be a DC slice part, a Network slice part, or an edge part, as well as the services running in each slice. Figure 16 presents the relationship between the model entities. The database is used by the Orchestrator for making its decisions. This model is described in detail in Deliverable D4.1

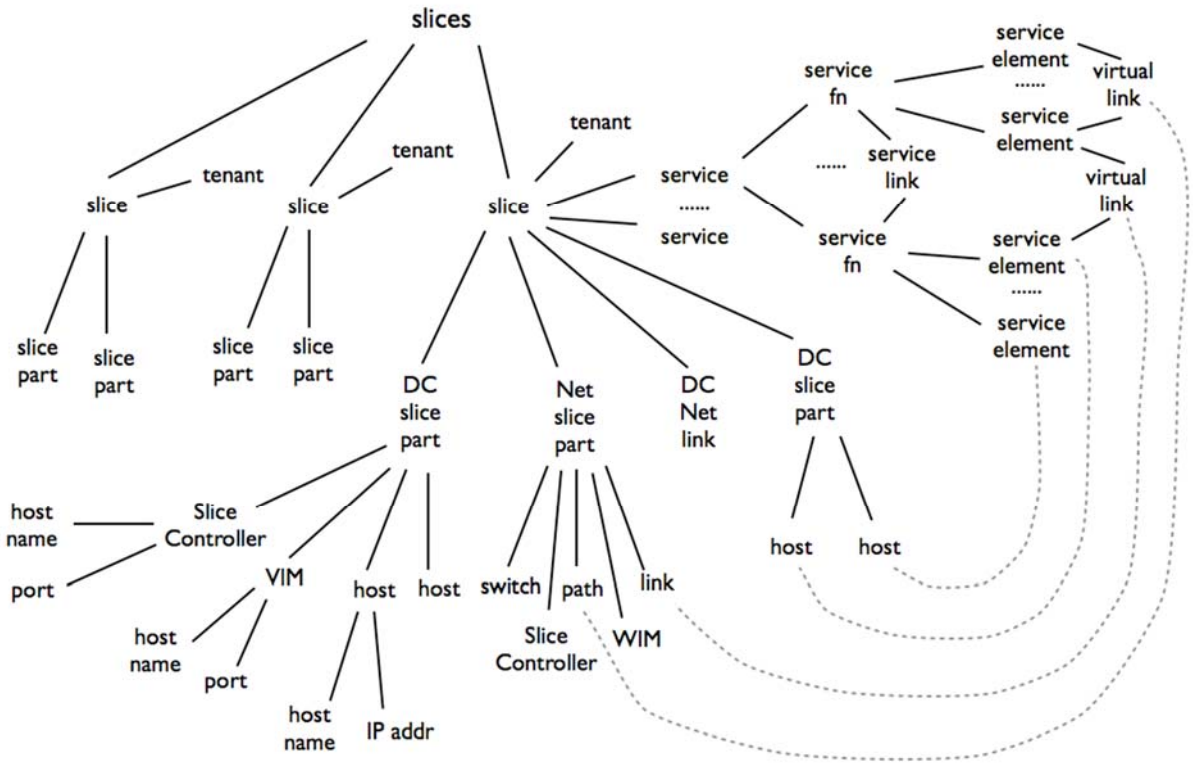

Figure 29. The Slice Database contains elements from the entity model

As we have highlighted in this document, a slice is made up of slice parts, along with one or more services are deployed into the slice.

The DC slice parts contain information including:

- The addresses of the VIMs, including their resource management endpoints and configuration / customisation endpoints
- The addresses of the DC Slice Controllers
- The addresses of the monitoring end-points
- Information about the hosts in the slice part

The Network slice parts contain information including:

- The addresses of the WIMs
- The addresses of the WAN Slice Controllers
- Information about the paths, links, or switches in the slice part

For each slice, the Slice Database will contain information on each of the services deployed into the slice. The information on the service will include:

- The tenant requesting and using the slice
- Each of the service functions in the service
- Each of the service links, which connect the service functions
- Information about each service element instance for the specific service function, whether it is a VM or a container, and the relevant instance attributes
- The embedding information for each service element, linking it to the specific host (these are shown as a dashed line in the figure).
- The virtual links, which are the instantiation of the service links that connect service element to service element.
- The embedding information for each virtual link.

The Slice Database also contains information about the starting times, adaption times, decommissioning times, etc. for all the relevant elements.

5.2 Infrastructure & Monitoring Abstraction

The Infrastructure & Monitoring Abstraction (IMA) component provides a uniform abstraction layer above the heterogeneous VIM / WIM and monitoring subsystems that are part of an end-to-end Slice. Different Slice Parts will constitute the end-to-end Slice, and each part can potentially rely on a different technology (e.g., specific VIM / WIM and monitoring subsystem implementations).

The aim of IMA is indeed to provide an abstract interface at its northbound to allow the SRO performing its functions without taking care of the implementation details of each Slice Part. In order to achieve that, multiple adapters will be allocated to hide the specific technological implementation of each Slice Part at the southbound of the IMA. The adaptors will in fact translate the requests coming through the northbound common API into the particular VIM or WIM API.

IMA will provide the SRO with an abstract way to perform the following functions while interacting with the underlying distributed resource domains that are part of the end-to-end Slice:

- Collecting information about the resource topology of each Slice Part.
- Collecting resource monitoring information belonging to each Slice Part (to be used for the Slice lifecycle management).
- Checking and verifying the status of the virtual elements allocated at each Slice Part (e.g., detecting failures associated to any of virtual service elements).

- Collecting resource facing KPIs (such as CPU, memory, storage consumption) for the virtual service elements running in the Slice Parts.

IMA consists of three main components, each one dealing with the implementation of the abstraction mechanisms required to perform the functions from the above list: (i) Resource and VM Management, (ii) Resource and VM Monitoring and (iii) Adaptors for VIM/WIM Control & Monitoring Interface.

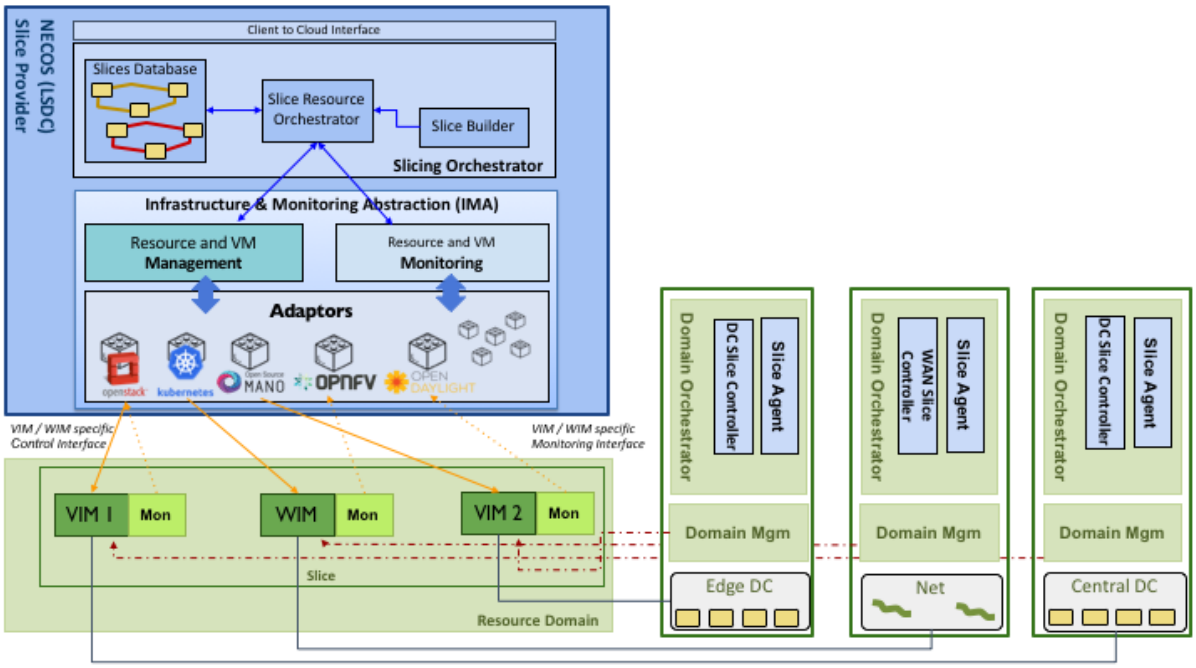

Figure 30. Infrastructure & Monitoring Abstraction (IMA) components

There is a separation on Management functions from Monitoring for two main reasons: (i) we observe that most VIMs are able to provide information related to the hosts, and on the embedding relationship of VMs to those hosts, but the VIMs only have a basic monitoring capability; (ii) to do proper slice and service function monitoring, we need a 'proper' monitoring system in order to do the extra functionality, beyond fundamental values from VIMs are needed. As it is too hard and complex to update every VIM, and as it is poor with respect to a separation of concerns, we decided that it is better to have a bespoke monitoring system.

In this view, the IMA needs to identify the type of VIM/WIM and its level of abstraction, in order to efficiently support and map requests with the appropriate wrapper. For example, the ability of a VIM/WIM to interact directly with the DC/network elements might provide more refined control on the provisioning and management of the requested resources. The identification and classification of VIM/WIN capabilities will be signalled/configured during the registration process.

We assume that a handle to a monitoring system (that was e.g., deployed on demand on the Slice part together with a VIM) is in fact returned back for each slice part to the Slice Builder (and the SRO) as soon as an end-to-end Slice is allocated. The handles to these monitoring subsystems will be used by the Resource and VM Monitoring IMA sub-component to gather the 'proper' more complete resource monitoring information flow; the Resource and VM Management IMA sub-component will instead interact with the VIM via the provided handle to perform resource management and gather fundamental information about the structure of the slice part and to collect an overview on the status of the related embedded virtual elements.

All the management and monitoring functionalities must be performed in such a way to provide isolation between monitoring and management data from different slices, i.e., a slice owner cannot see other's slices data. This applies both for the infrastructure and service level monitoring and management.

5.2.1 Resource and VM Management

The Resource and VM Management stands as the IMA sub-component responsible for providing the abstract API to be used by the SRO to interact with the VIM / WIM inside the different Slice parts to perform both resource operations and VM management operations. The SRO will use generic methods to collect the infrastructure view of each Slice part in order to build the global view of the end-to-end Slice, on which the service elements requested by the tenants will be embedded.

This component will receive requests to gather data related to the status of the physical resources and their relationship, together with a map of how virtual elements are placed on the resources. This gives a global view which can be represented as a structure, e.g., a tree of physical and virtual elements. This data is different from the real-time monitoring information flow that will be gathered by the Resource and VM monitoring sub-component, as it only provides a way to the SRO to check that the embedding process is consistent with the status of the virtual elements actually running on the Slice parts. More specifically, this entry point will be used to:

- retrieve an up-to-date topology view of the Slice part that will be used to build the global resource view of the end-to-end Slice;
- check the status of the virtual elements allocated on a Slice Part for each embedded service instance (e.g., to detect any failures on the VIM).

This relationship data does not give any real-time information, as such flows are handled by the Resource and VM Monitoring component.

In a similar way, the SRO will use generic methods to perform the lifecycle management of services, which includes the instantiation and removal of virtual elements (VMs, containers, virtual switches, and virtual routers), regardless of the VIM implementation of that slice part and the specific related API (e.g., Openstack API rather than Docker API, etc.).

5.2.2 Resource and VM Monitoring

The Resource and VM Monitoring IMA sub-component implements the set of abstractions required to provide the SRO with a homogeneous view on the resource utilisation using monitoring information gathered from the different Slice parts. The real-time data gathered from the monitoring is directly associated with the structured global view gathered by the Resource and VM Management component. This type of monitoring information complements the resource state information collected by the Resource and VM Management components – while the latter provides feedback about the status of the virtual resources deployed on the slice (mainly to detect faults and check the consistency between the performed embedding operations and the actual run-time resource status), the former provides the actual real-time monitoring flows that will be used to check the performance of the physical and virtual resources elements of the slice. By combining both the structure and the real-time flows, we get a clear view on the current status of all slice elements.

In order to implement the above-mentioned abstractions, the Resource and VM Monitoring IMA sub-component interacts with both the VIM / WIM entry point and with the monitoring entry point that were returned as relevant element of that Slice part. More specifically, this component can be used to collect the resource facing KPIs (CPU, Mem and Storage) of both the physical resource and the Virtual service elements running on that VIM / WIM.

The collected information will then be passed on to the Service Orchestrator, to perform the lifecycle management of the service instances. This information will complement the Service Specific KPIs already collected by the Tenants via its external monitoring loop – Service Level Monitoring.

The monitoring entry-point of a specific Slice part will be used to collect monitoring information related to the usage of the resource entities of that Slice part. As each monitoring entry-point might be related to a specific type of monitoring subsystem that was instantiated on that Slice part, an abstracted way to collect the information and a coherent representation of it should be provided by the Resource and VM Monitoring subcomponents to the Slice Resource Orchestrator. This information will then be used by the SRO to perform the lifecycle operation of the Slice.

5.2.3 Adaptors for VIM/WIM Control & Monitoring Interfaces

Specific adaptors for particular VIMs and WIMs are required to support multiple technologies usually providing different levels of abstraction and a variety of supported features.

The VIM and WIM Adaptors are specific wrappers for different VIM/WIM implementations, providing a basic wrapping-agnostic service platform to common NECOS functions. In practice, the adaptors should translate generic calls into specific commands or methods in the context of specific VIMs and WIMs. Typical VIM and WIM adaptors include:

- Cloud adaptors: Openstack, Heat, Kubernetes, VLSP;
- VNF adaptors: OpenMANO, Open Baton, OPNFV;
- Transport adaptors: SDN controllers, such as Opendaylight, Floodlight;
- RAN adaptors;
- Edge adaptors.

Additional adaptors for the monitoring could be employed in the sense that there are different technological solutions for collecting monitoring data. In this sense, local monitoring systems should have adaptors so that the monitoring information is gathered in a way compliant to the information model defined in D4.1.

5.3 Resource Domains

In order to support service provisioning over the slices, a mechanism capable to afford the slicing of the DC compute and storage resources, as well as the network resources, is necessary. To manifest this slice approach in NECOS, two components at the infrastructure level act as a point of control and management of DC and Network slices. For each data center, a DC Slice Controller is in charge of creating DC slices within the data center and allocating the required compute and storage resources. This process can be different according to the level of abstraction used by a tenant for a Slice request, and the type of mode the NECOS system is asked to operate (see Section 2.4 for further details). The resource allocation can end-up with the deployment of an on-demand VIM based on a particular tenant specification or the creation of separate handle for a Slice in a shared VIM to be passed back to the requesting tenant. Similarly, for each network domain, a WAN Slice Controller is in charge of instantiating a network slice between two DC slices and deploying the on-demand WIM or a reference to a Slice created in an existing WIM instance. In the following, we provide a brief description of both the DC and WAN Slice Controller components.

5.3.1 DC Slice Controller

The DC Slice Controller component resides in each Data Center Provider, being responsible to dynamically create a data center slice. To accomplish this task, the DC Slice Controller performs the following operations:

- *Resource Management*: the DC Slice Controller manages a pool of all of the DC (compute and storage) resources in the data center that are allocated to participate in slicing and keeps track

of which resources have been allocated to which slice, together with meta-data such as the VIM entry point, and the keys used for access to all the data center resources.

- *Slice Creation*: the DC Slice Controller handles DC Slice creation requests for DC slices coming from the Slice Agent, and determines if it is possible to create a new DC slice in the data center based on local resource availability.
- *Resource Allocation*: if the DC slice creation is possible, the DC Slice Controller is responsible for selecting the DC resources, from the pool, that should be allocated to the slice.
- *VIM Deployment/Shim deployment*: In Mode 0, for each DC slice there is an on-demand VIM. In this case, the DC Slice Controller is responsible for allocating and deploying a VIM of a particular type to the DC slice, and configure it to use the DC resources which have been picked for the slice. This is achieved by creating VIM template images that are stored in an image repository, and adapted according with the slice specification. In Mode 1, for each data center, there is an existing VIM running in that Resource Domain. In this case, the DC Slice controller is responsible for creating an isolated interaction point (such as a shim) for the tenant.
- *Slice Elasticity*: under the control of the Slice Resource Orchestrator, the DC Slice Controller updates (ads or removes) the DC resources assigned to a slice on-the-fly as a slice can grow or shrink at runtime.
- *Slice Deletion*: the DC Slice Controller handles requests for the deletion / shutdown of DC slices. The resources used by the slice will be returned to the resource pool. In the Mode 0 case, the VIM allocated for the slice will also be shutdown.
- *Slice to Network Connectivity*: the DC Slice Controller is responsible for connecting an allocated DC slice part to a specified external network end-point. This will create a special tunnel connecting the VLAN of the DC slice part to the specified external network end-point.

5.3.2 WAN Slice Controller

The WAN Slice Controller is the component that resides in each Network Provider to dynamically creating a Network slice. A Network slice is a virtual networking infrastructure (e.g., nodes, links, functions, etc.) necessary to properly connect two DC slices. In order to create a Network slice, the WAN Slice Controller performs the following operations:

- *Network Management*: the WAN Slice Controller manages all of the network resources in the network provider domain that are allocated to participate in slicing, and keeps track of which network resources have been allocated to which slice.
- *Slice Creation:* the WAN Slice Controller handles requests for Network slices coming from the Slice Agent, and determines if it is possible to create a new network slice in the network domain according with based on local resource availability (e.g. bandwidth).
- *Resource Provisioning*: if the Network slice creation is possible, the WAN Slice Controller is responsible for providing the virtual networking infrastructure required to best connect the given DC slices (i.e., meeting the network slice requirements).
- *WIM Deployment/Shim deployment*: In mode 0, for each Network slice there is an on-demand WIM. In this case, the WAN Slice Controller deploys a WIM to the Network slice, and configures it to use the network resources which have been assigned for the slice. In Mode 1, for each network domain, there is an existing WIM running in that domain. In this case, the WAN Slice controller is responsible for creating an isolated interaction point (such as a shim) for the tenant.
- *Network Slice Elasticity*: under the control of the *Slice Resource Orchestrator*, the WAN Slice Controller updates the network resources assigned to the slice on-the-fly, as a slice can grow or shrink at runtime.

- *Slice Deletion:* the WAN Slice Controller allows handles requests for the deletion/shutdown of WAN slices. All resources previously allocated will be returned to the resource pool, and the WIM allocated for the slice will also be shutdown.

5.4 Marketplace

The *Marketplace* is a supporting fully-distributed system that locates suitable slice parts from a set of participating resource domains. As mentioned, the *Slice Builder* will interact with a *Slice Broker* in terms of triggering a Slice request, and receive a list of suitable slice parts in response. The *Slice Broker* discovers these slice parts by interacting with a set of *Slice Agents* that are hosted by the involved resource domains.

Due to the distributed nature of the Marketplace, it is possible to have multiple Marketplace instances. We view the Marketplace in NECOS in a similar way to existing online marketplaces, such as those for hotels (e.g., booking.com) or for flights (e.g., skyscanner.net), yielding hopefully the same business opportunities for the participating resource providers.

Within NECOS, we consider the following kinds of Marketplace:

- *the telecoms Marketplace*, which is a limited setup between close co-operating telecoms providers, i.e., allowing the creation of slices across each other's infrastructure,
- *the contract Marketplace*, which is a setup between partners with a level of trust based on signed contracts, i.e., allowing their resources to be used by others for slices,
- *the open Marketplace*, which allows any provider to offer resources and register their offerings for users.

The internal operation between the *Slice Broker* and the *Slice Agents* is not exposed externally, but we propose the following two models of operation:

- a *pull model* – where the *Slice Broker* interacts with the *Slice Agent*, and the *Slice Agent* dynamically determines the availability with the resource domain, at run-time,
- a *push model* – where the resource provider pre-determines which resources and groups of resources to be allocated for slices, and these are pushed out to a catalogue in the *Slice Broker*.

The mechanisms to realize the above operation models are detailed in deliverable D5.1, while a resource discovery workflow with an information model representation of the requested and offered resources in deliverable D4.1. We give a functional description of the *Slice Broker* and the *Slice Agent* components below.

5.4.1 Slice Broker

The *Slice Broker* is the central component of a Marketplace. It accepts requests for slices, and is responsible for locating resource providers that are able to satisfy the requests, based on a set of constraints and requirements, such as amount of resources, location, as specified in deliverable D4.1.

In order to realize the resource discovery process, the *Slice Broker* decomposes the original request to a set of slice part requests, and addresses the requests to a number of *Slice Agents*, each belonging to a particular resource domain. For each slice part, for instance a requirement for N hosts in a particular country, the *Slice Broker* may enquiry multiple *Slice Agents* whether they can satisfy that request. The answers collected for each slice part present alternative options for the slice resources to be instantiated to create the slice. Alternatives for each slice part will be returned as a reply to the request originating from the *Slice Builder*. The *Slice Builder* will select the resources for each slice part based on a relevant decision mechanism (i.e., considering the cost model and the suitability of the resources), as described in Section 5.1.4.

This discovery process takes place in two main stages. In the first stage, the *Slice Broker* contacts the *DC Slice Agents* participating in the particular marketplace, and collects alternative computing resources for each slice component. Based on the previous information, in the second phase, the *Slice Broker* requests from the marketplace's *WAN providers* network connectivity options to interconnect the alternative computing resources. This targeted query is based on the slice structure, i.e., the latter indicates the connectivity requirements for the alternative computing slice resource components. The information obtained for the DC and WAN are processed by the Slice Builder, which proceeds with a filtering mechanism for alternatives: alternatives concerning computing resources that cannot be "connected" via network resources are naturally discarded from the *Slice Broker's* reply, since they cannot possibly appear in the final slice instantiation.

However, it might be the case that a slice could not be formed based on the tenant's requirements. For instance, consider the case that a tenant requests a single computing resource with an excessive amount of memory; in this case the negative reply to the *Slice Builder* must include some information regarding the cause of failure in the discovery process. This information is useful for the *Slice Builder* to change (if possible) the slice requirements and submit a new request for the slice creation.

Finally, it should be mentioned that each *Slice Broker*, depending on the kind of Marketplace it realises will have different registration policies for *Slice Agents*. Consider the following two extremes cases: on one end is the Telecoms Marketplace, with a closed list of provider *Slice Agents* available, and on the other end the Open Marketplace, offering open access to providers, i.e., they can dynamically join or leave the marketplace. In the second option, the providers can financially exploit their excessive resources in a more dynamic manner. This flexibility of the NECOS approach allows for the accommodation of a variety of federation schemes, i.e., from tightly to loosely coupled.

5.4.2 Slice Agent

The *Slice Agent* component resides at each infrastructure provider. It is responsible for determining if a slice part request can be satisfied, and responds with resource volumes, a cost model and possible SLA/SLO attributes. In order to reply to the *Slice Broker's* requests, the *Slice Agent* communicates with its local DC/WAN domain controller, i.e., offering updated information on the available resources. Thus, the *Slice Agent* is a "translation" component that allows local domain controllers to expose resource information to the marketplace. This allows NECOS to interoperate with a diverse range of controllers, given that technology specific *Slice Agents* will be realised. In other words, a typical *Slice Agent* implementation abstracts and translates the offered resources from heterogeneous domain controllers in the NECOS information model representation (i.e., defined in deliverable D4.1), using a technology-agnostic north interface, the *Slice Marketplace Interface*, and a technology-dependent south interface.

It should be mentioned that a *Slice Agent* can participate in multiple Marketplaces, i.e., it can register and receive requests from multiple *Slice Brokers*. This set of Marketplaces can range over all kinds mentioned above, i.e., can be associated with multiple Telecoms or Open Marketplaces. In this setting, the local *Slice Agent* can incorporate policies regarding the information resource providers are willing to share, depending on the type of the marketplace the request originates from. Again considering the two extremes, in the Telecoms Marketplace, since the latter is a federation of closely cooperating providers, a *Slice Agent* might be willing to share information, whereas in the Open Marketplace, the Agent can be more cautious on releasing sensitive data. The same applies regarding requests for resources: in the Telecoms case, the *Slice Agent* will always reply positively to a request given that it has available resources, whereas in the Open Marketplace a resource provider might implement different business policies on replying to incoming requests, i.e., may prioritize offering the resources to its associates participating in a Telecoms Marketplace. Thus, the providers can implement bespoke resource offering policies through customized *Slice Agent* implementations, aligned to the applied federation scheme, resource utilisation levels and the chosen business model.

We believe the marketplace concept is a NECOS asset that demonstrates adequate flexibility to accommodate a range of federation schemes through allowing the implementation of alternative

provider resource offering policies addressing the diversity of cooperation situations in the current cloud/network landscape.

6 Interactions between the functional blocks

This section presents the interactions between the functional blocks within the NECOS architecture.

6.1 Slice lifecycle

Slice is a dynamic entity that has its own lifecycle management and operation. As depicted in Figure 18, the lifecycle management typically includes an instantiation, configuration and activation phase, a run-time phase and a decommissioning phase. A preparation phase is also defined and comprises all the tasks that are performed before the slice instance is created.

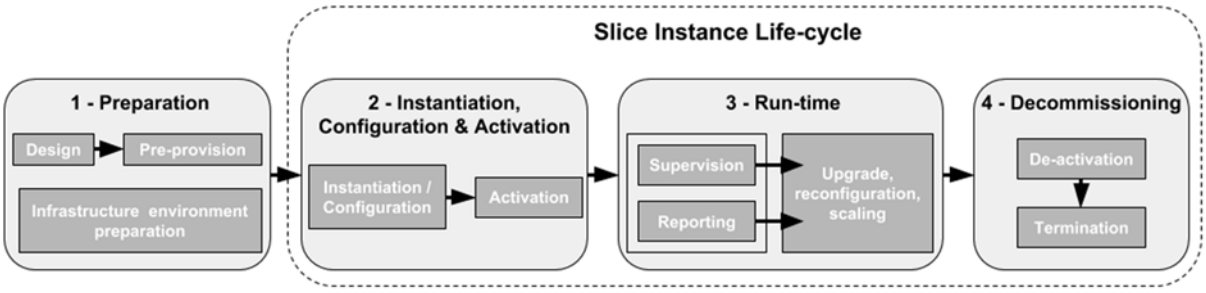

Figure 31. Lifecycle phases of a slice instance. (Source: adapted from [TR28801])

In NECOS platform the main tasks performed by each of these four phases are:

1. Preparation Phase: this starts with the specification of a slice using slice templates, and is performed by the Tenant; it is followed by the discovery and pre-provisioning of resources involving the Slice Builder, Slice Broker and Slice Agents via the Slice Request Interface and Slice Marketplace Interface; continues with the preparation of the allocated infrastructure for the instantiation and support of the slice performed by the Slice Builder interacting with the Slice Controllers via the Slice Instantiation Interface.
2. Instantiation, Configuration and Activation Phase: as soon as all the resources shared/dedicated to the slice instance have been created and are configured, the Slice Resource Orchestrator will complete and activate the Slice using the handles to the different Slice Parts received by the Slice Builder.
3. Run-time Phase: normal operation of the slice and monitoring of its performance (via the IMA), allowing the Slice Resource Orchestrator to perform the slice's reconfiguration or scaling as necessary using the Slice Runtime Interface.
4. Decommissioning Phase: deactivation of the slice, releasing its allocated resources is carried by the Slice Resource Orchestrator at the end of the Slice lifetime via the Slice Runtime Interface.

In the NECOS platform, each phase of the slice lifecycle will trigger actions on different components. This tasks should be coordinated by the Slicing Orchestrator in order to provide ways of handling workflows in scenarios of success or failure. Figure 19 illustrates in detail which components have tasks assigned in each of the lifecycle phases as introduced in the aforementioned description.

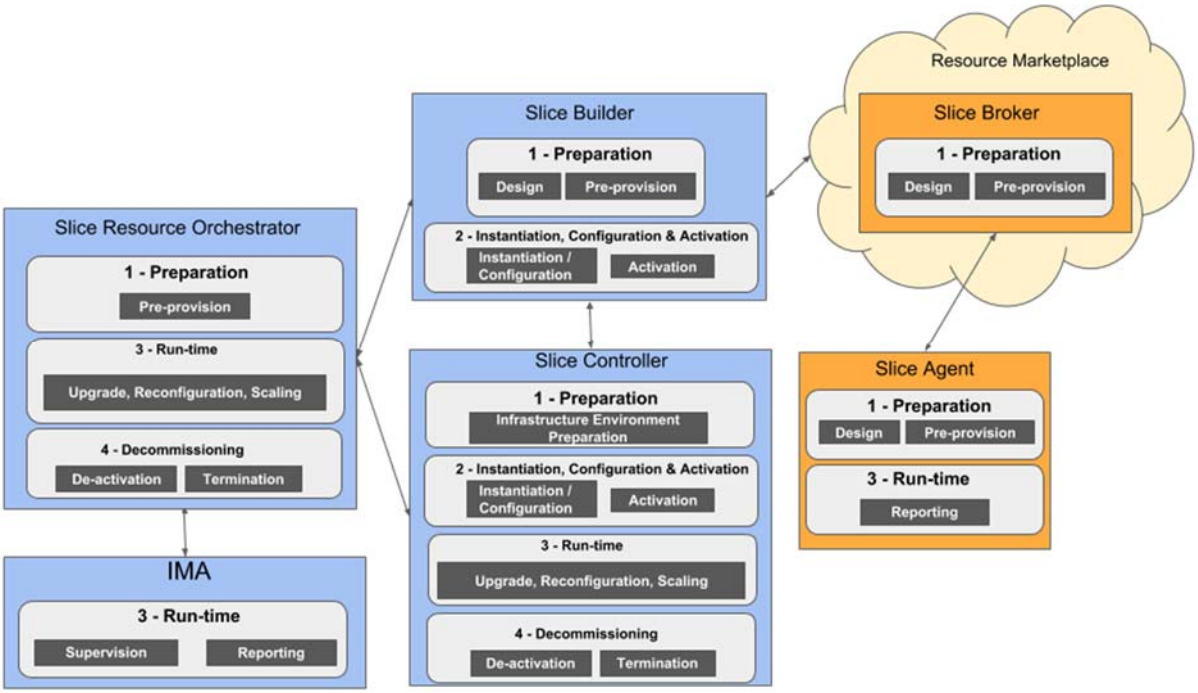

Figure 32. Mapping of slice lifecycle operations and NECOS components

Each one of these lifecycle phases is further broken down into sub-phases, in which a specific workflow will be performed by different components, as explained in the remainder of this Section. The operations related to each sub-phase will also be linked to the numbered steps (reported in square brackets) of the overall workflow described in Section 6.2.1 and depicted in Figure 21 (e.g., Design sub-phase in the Preparation is related to step 1 of the overall workflow).

6.1.1 Preparation phase

This phase starts with the specification of slice templates, followed by the discovery and pre-provisioning of resources, then the preparation of the allocated infrastructure for the instantiation and support of the slice, and the exchanging of agreement between tenants and providers. It is divided into three sub-phases:

- **Design**
  The Slice Builder uses the slice blueprints/descriptors to design a new slice instance. Here the different slice parts are described as well as their features, configurations, resources needed and associated workflows. [Step 1]
- **Pre-provision**
  The Slice Builder together with the Slice Broker and the Slice Agents makes an on-demand request of resources via the Slice Request and Marketplace Interfaces using admission control, resource negotiation and charging. Network resource availability, latency and resiliency, as well as polices, are taken in consideration before doing the allocation. [Steps 2,3,4,5,6,7]
- **Infrastructure environment preparation**
  The Slice Builder uses the Slice Instantiation Interface to prepare all the infrastructure resources needed for the instantiation and support of the new slice to be created, via contacting the relevant DC or WAN Slice Controllers, with the *offer* from the related Slice agent. [Step 8]

6.1.2 Instantiation / configuration and activation phase

In this phase all resources of the slice instance are created and configured and become ready for operation. These tasks occur in the Resource domains.

- **Instantiation/Configuration**
  The Slice Controller finalises the allocation of the resources for the Slice: the required infrastructure resources (shared and/or dedicated) are configured and instantiated but not yet activated. Allocating the VIMs. [Steps 9,10]
- **Activation**
  The Slice Resource Orchestrator finalises the activation of the slice performing the steps required to complete its topology; any pending service deployment is processed, service instances are activated and becomes ready to start their operation. [Steps 11,12,13,14,15,16]

6.1.3 Run-time phase

In this phase, the slice instance is normally operating and has a feedback loop provided to the SRO via the IMA that is used to monitor the slice performance as well as the overall infrastructure needs.

- **Supervision & Reporting**
  The performance and status of the Slices are monitored and reported to the SRO through the IMA component. According to the requirements of each slice and also to the overall status of the system, the SRO can take the decision of performing reconfiguration and / or scaling on (some of) the running slices.
- **Upgrade, Reconfiguration, Scaling**
  If required, a slice instance is re-configured and / or scaled according to its specific requirements and to the overall status of the system. This process allows performing resource optimisation via slicing elasticity and is further detailed in the remainder of this Section.

6.1.3.1 Scaling and elasticity

Supported by the monitoring data, the management of the slice instance involves strategies to provide elasticity, adding or removing resources from the slice according to its workload, ensuring optimized service performance with efficient resource usage. These methods may be applied both by the NECOS Slicing Orchestrator, to add/remove or replace resources allocated to a given slice.

Here we present more of the options for Scaling a slice and presenting slice elasticity.

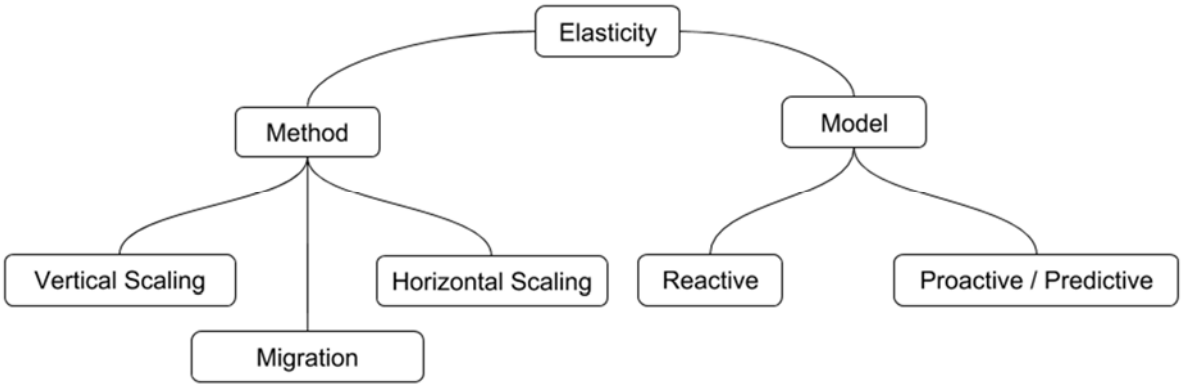

Figure 33. Categorization of methods and models of elasticity solutions

In the 3GPP approach there are 2 methods for scaling. We have adapted these for NECOS:

- **Horizontal Scaling**
  This scaling is appropriate when new Slice parts are needed or when an existing Slice part can be shutdown. To add a new Slice part, the Slice Builder / Slice Broker mechanism needs to be triggered to find the relevant part. When doing a shutdown of an existing part, the Slice Resource Orchestrator can use the Slice Runtime Interface to tell the relevant DC Slice Controller to close the Slice part.
- **Vertical Scaling**
  This scaling is appropriate for resources in existing Slice parts. It involves adding/removing resources from a slice part (e.g. computing, memory or storage) or network part. The NECOS Slicing Orchestrator might add/remove physical devices or resources from a slice part to better support its current demand. This is achieved by the Slice Resource Orchestrator using the Slice Runtime Interface to make a request to the relevant DC Slice Controller.
- **Migration**
  For a slice, migration needs to be analysed and defined.

The models are:

- **Reactive**
  The elasticity procedures are triggered by thresholds in resource usage or SLA violations, reacting to the current workload.
- **Proactive / Predictive**
  Uses predictive mechanisms, possibly based on workload history, to trigger the elasticity procedures according to anticipated load.

6.1.4 Decommissioning phase

After a slice instance is no longer needed, the Slice Resource Orchestrator proceeds to finish the lifecycle of the slice.

- **De-activation**
  The slice instance is deactivated, stopping all its operations and services.
- **Termination**
  The slice instance is terminated and all the allocated resources are released and reclaimed.

6.2 Workflows and interactions

This section presents the interactions between the functional blocks.

It will use the named APIs between the components and uses the named interfaces and APIs being called:

- *The Slice Request Interface:* provides the mechanisms to the Slice Builder to initiate the instantiation of an end-to-end Slice via interacting with the Slice Broker in the Resource Marketplace. It is used by the Slice Broker to provide the description of the Slice to be allocated.
- *The Slice Marketplace Interface:* is related to the interaction between the Slice Broker and the Slice Agents to implement mechanisms for the propagation of resource offerings between (external) resource domains. Different Slice Agents will register their resources availability to the Slice Broker.
- *The Slice Instantiation Interface:* is used by the Slice Builder after available resources offerings have been identified via the Slice Broker; individual resource allocation requests are sent to the

relevant Slice Resource Controllers which allocate resources for each single Part of the end to end Slice.

- *The Slice Runtime Interface:* implements the interface that provides functionalities to dynamically modify the resource allocation for a Slice Part. This is required by the Slice Resource Orchestrator to perform lifecycle operation on the end to end Slice according to the feedback received for each Slice Part via the collected monitoring measurements.

6.2.1 Preparation and instantiation workflow

The lifecycle flow for the preparation and instantiation is shown below. More details of all the lifecycle functions are presented in Section 6.1.

The steps for preparation of a slice plus it's instantiation are presented in the following list:

1. The end-user makes request to a Slice Activator for a Slice using a *Slice Description*, with a specification for the slice, either as a resource focussed spec or as a service focussed spec with some service level details
2. The Slice Activator makes a request to the NECOS Slice Provider handled by the Slice Specification Processor.
3. The NECOS Slice Provider asks Slice Builder to create Slice elements based on the specification
4. The Slice Builder goes to the Marketplace of choice, and interacts with a Slice Broker, to find some slice parts.
5. The Slice Broker interacts with a number of Slice Agents, to try to find suitable and available Slice Parts
6. The Slice Broker collects the details of the infrastructure provider's offers from each of the Slice Agents

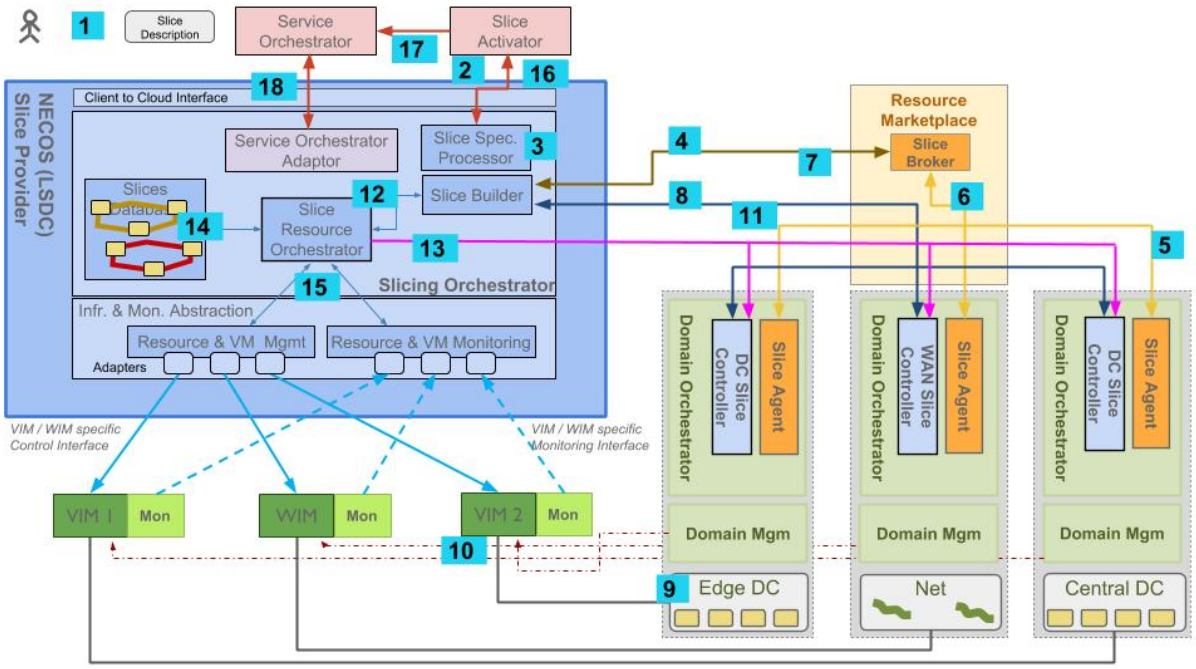

Figure 34. Steps for the preparation and instantiation workflow

7. The Slice Builder receives these details from the Slice Broker in the Marketplace, and makes a decision as to which offers to choose for the Slice
8. The Slice Builder contacts all the relevant infrastructure Slice Controllers with a request for a slice, plus some specifications
9. The Slice Controllers allocate the relevant resources.
10. The Slice Controllers allocate / instantiate / configure the on-demand VIM and WIM to manage the resources of the allocated Slice part, as well as the relevant monitoring end-points.
11. The Slice Controllers return the details of the Slice part, together with the access details for the VIM or WIM, plus the associated monitoring end-point.
12. The Slice Builder returns all the Slice Controller details plus Slice component details (e.g. VIMs) to the Slice Resource Orchestrator
13. The Slice Resource Orchestrator interacts with the DC/WAN Slice Controllers to bind/glue the separate slice parts together. This completes the topology as it connects DC part to Net part.
14. The Slice Resource Orchestrator keeps a representation of the slice topology and its constituent parts.
15. The Slice Resource Orchestrator informs the IMA of the allocated VIMs, WIMs, and monitoring end-points, and then the IMA sets up the relevant Adaptors for the VIMs, WIMs, and monitoring end-point at the resource domains.
16. The Slicing Orchestrator informs the Slice Activator of the details of the slice.
17. Slice Activator triggers the Service Orchestrator and the deployment of service is started into the Slice
18. The VMs are instantiated

6.2.1.1 Geographic viewpoint

In the following figures we show a geographical viewpoint, which demonstrates the distributed nature of the NECOS cloud systems, and how each of the steps of this workflow have an interaction with various systems, with different roles, in multiple locations.

These figures present a scenario of a user in Italy, who wishes to setup a slice and deploy service elements in Spain, Greece, and the UK, with a virtual network connecting the service elements. To activate this, the user contacts a tenant of the NECOS system, who has their own Service Orchestrator and own Slice Activator. This tenant uses a Slice Provider that is in Italy.

There are multiple DCs in Spain, Greece, and the UK, and Network providers in France, Spain, and Germany, which are candidates for offering resources for a slice. The use of the Marketplace finds the resource for the slice at run-time.

Step 1 - The end-user makes request to a Slice Activator for a Slice using a *Slice Description*, with a specification for the slice, either as a resource focussed spec or as a service focussed spec with some service level details

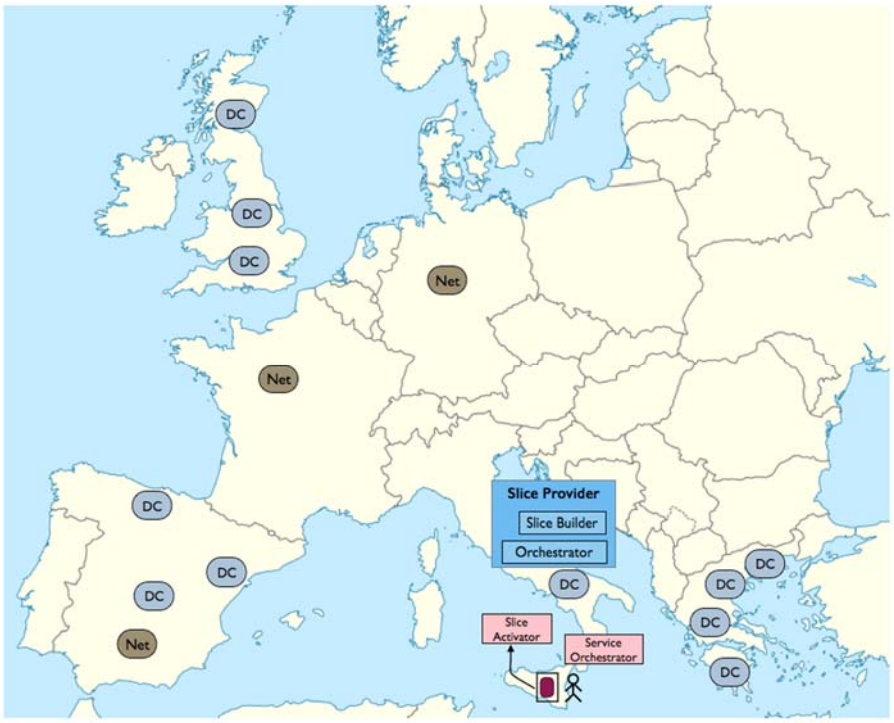

Figure 35. Step 1 of the preparation and instantiation workflow

Step 2 - The Slice Activator makes a request to the NECOS Slice Provider. The request is passed onto the Slice Spec Processor for analysis and processing.

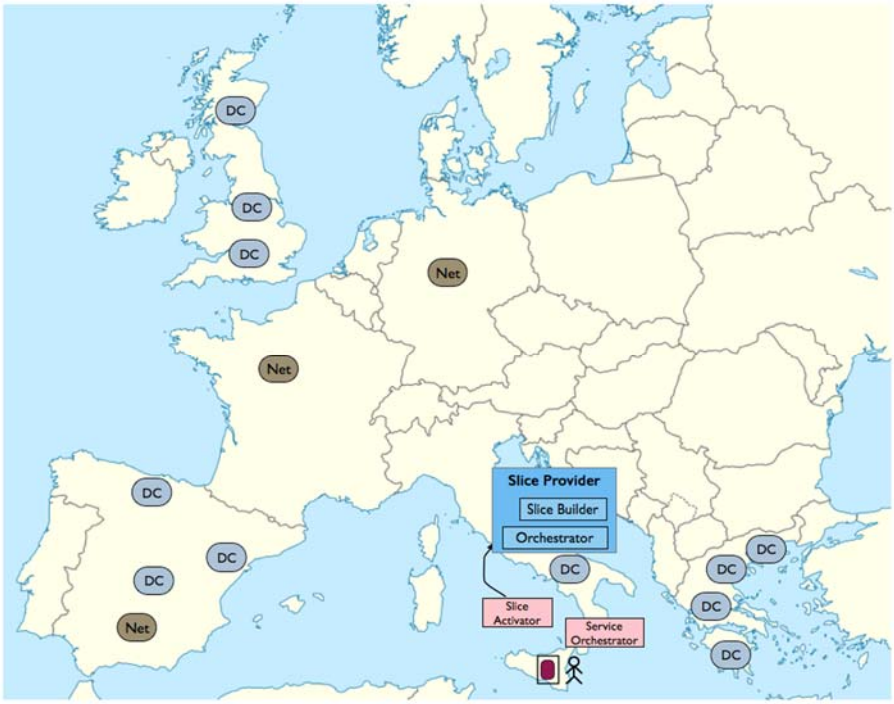

Figure 36. Step 2 of the preparation and instantiation workflow

Step 3 - NECOS Slice Provider asks Slice Builder to create Slice elements based on the specification

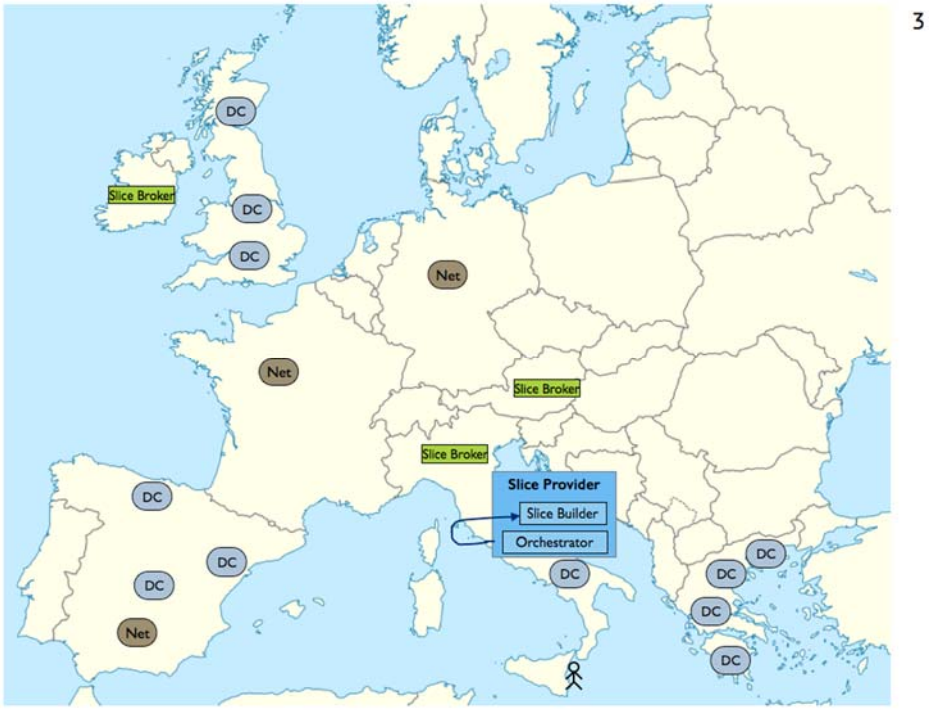

Figure 37. Step 3 of the preparation and instantiation workflow

Step 4 - The Slice Builder goes to the Marketplace, and interacts with a Slice Broker, to find some slice parts.

Figure 38. Step 4 of the preparation and instantiation workflow

Step 5 - The Slice Broker interacts with a number of Slice Agents, to try to find suitable and available Slice Parts

Figure 39. Step 5 of the preparation and instantiation workflow

Step 6 - The Slice Broker collects the details of the infrastructure provider's offers from each of the Slice Agents

Figure 40. Step 6 of the preparation and instantiation workflow

Step 7 - The Slice Builder receives these details from the Slice Broker in the Marketplace, and makes a decision as to which offers to choose for the Slice

Figure 41. Step 7 of the preparation and instantiation workflow

Step 8 - The Slice Builder contacts all the relevant infrastructure Slice Controllers with a request for a slice, plus some specifications

Figure 42. Step 8 of the preparation and instantiation workflow

Step 9 - The Slice Controllers allocate the relevant resources.

Figure 43. Step 9 of the preparation and instantiation workflow

Step 10 - The Slice Controllers allocate, instantiate, and configure the on-demand VIM and VIM to manage the resources of the allocated Slice part, as well as the relevant monitoring end-points.

Figure 44. Step 10 of the preparation and instantiation workflow

Step 11 - The Slice Controllers return the details of the the Slice part, together with the access details for the VIM or WIM, plus the associated monitoring end-point.

Figure 45. Step 11 of the preparation and instantiation workflow

Step 12 - The Slice Builder returns all the Slice Controller details plus Slice component details (e.g. VIMs) to the Slice Resource Orchestrator

Figure 46. Step 12 of the preparation and instantiation workflow

Step 13 - The Slice Resource Orchestrator interacts with the DC Slice Controllers to bind/glue the separate slice parts together. This completes the topology as it connects DC part to Net part.

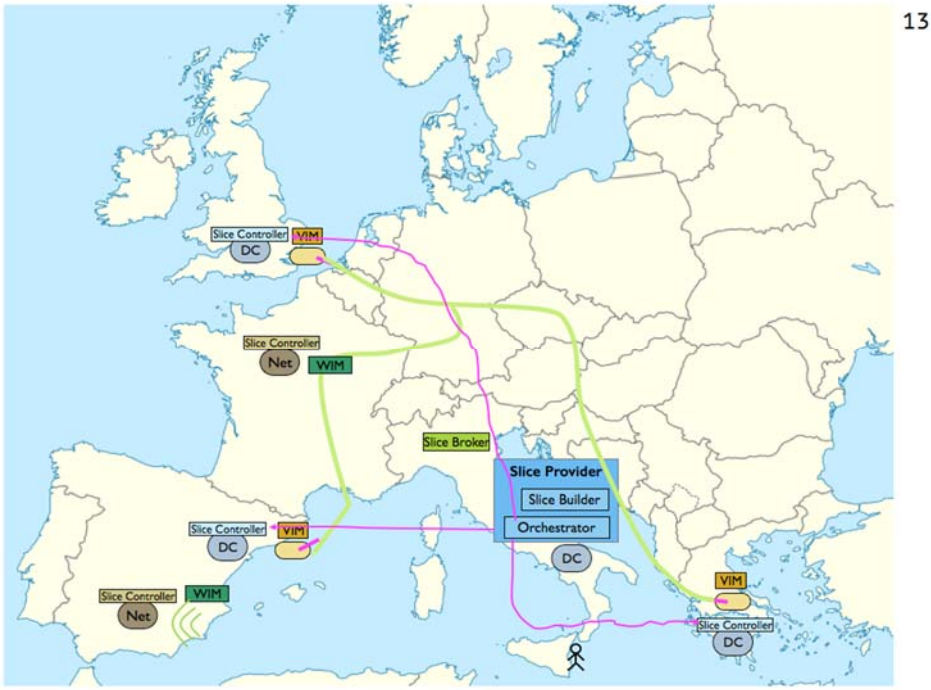

Figure 47. Step 13 of the preparation and instantiation workflow

Step 14 - The Slice Resource Orchestrator keeps a representation of the slice topology and its constituent parts.

Figure 48. Step 14 of the preparation and instantiation workflow

Step 15 - The Slice Resource Orchestrator informs the IMA of the allocated VIMs, WIMs, and monitoring end-points, and then the IMA sets up the relevant Adaptors for the VIMs, WIMs, and monitoring end-point at the resource domains.

Figure 49. Step 15 of the preparation and instantiation workflow

Step 16 - The Slicing Orchestrator informs the Slice Activator of the details of the slice.

Figure 50. Step 16 of the preparation and instantiation workflow

Step 17 - Slice Activator triggers the Service Orchestrator and the deployment of service is started into the Slice

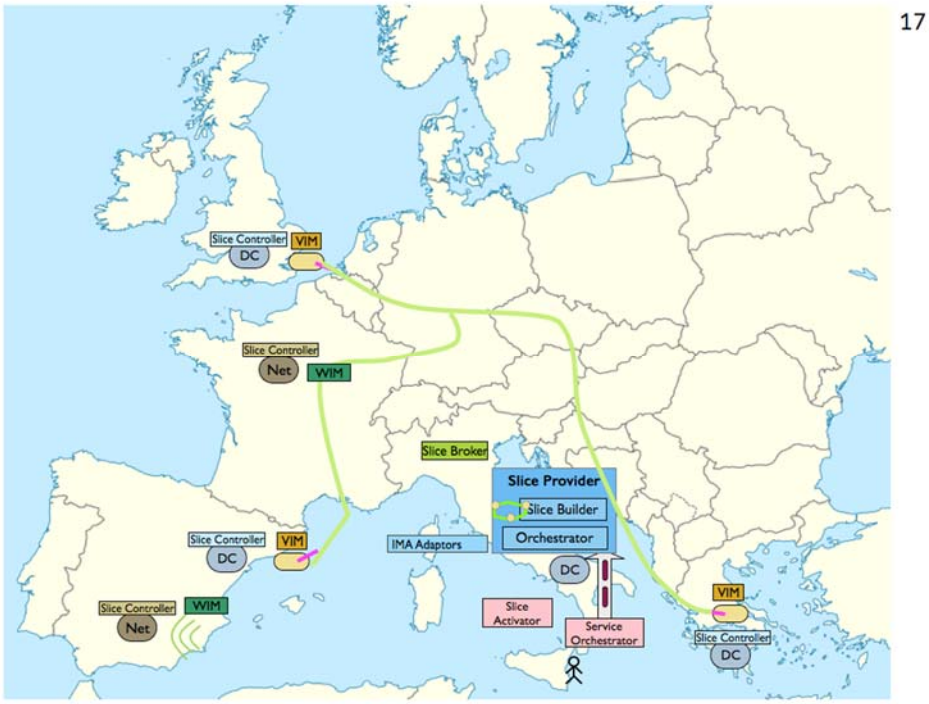

Figure 51. Step 17 of the preparation and instantiation workflow

Step 18 - The VMs are instantiated

Figure 52. Step 18 of the preparation and instantiation workflow

The service is now ready for access by the end-users.

6.2.2 Run-time and decommissioning workflow

The lifecycle flow for the run-time phase is shown below in Figure 40. More details of all the lifecycle functions can be found in the Slice Lifecycle section.

1. The Slice Resource Orchestrator (SRO) functional block is meant to continuously receive feedback sets from the Infrastructure Monitoring Abstraction (IMA). These feedbacks include relevant slice-specific KPIs (e.g., network load, utilization of allocated resources).
2. Lifecycle actions are triggered according with the IMA feedback, including automatic reconfiguration of slices or decommission process, which is carried out by SRO via the Slice Runtime Interface.
3. If a slice needs to add a new slice part in a new location or domain, the SRO contacts the Slice Builder to enforce the process of discovering new slice parts and attached them to the existent slice in a similar manner as described for the Slice Instantiation Phase.
4. If a slice needs to expand, the SRO contacts the relevant Slice Controllers in order to extend the slices parts, increasing the allocated resources associated to the slice parts.
5. If a slice needs to shrink, the SRO contacts the corresponding Slice Controllers in order to remove slices parts, releasing the allocated resources associated to slice parts

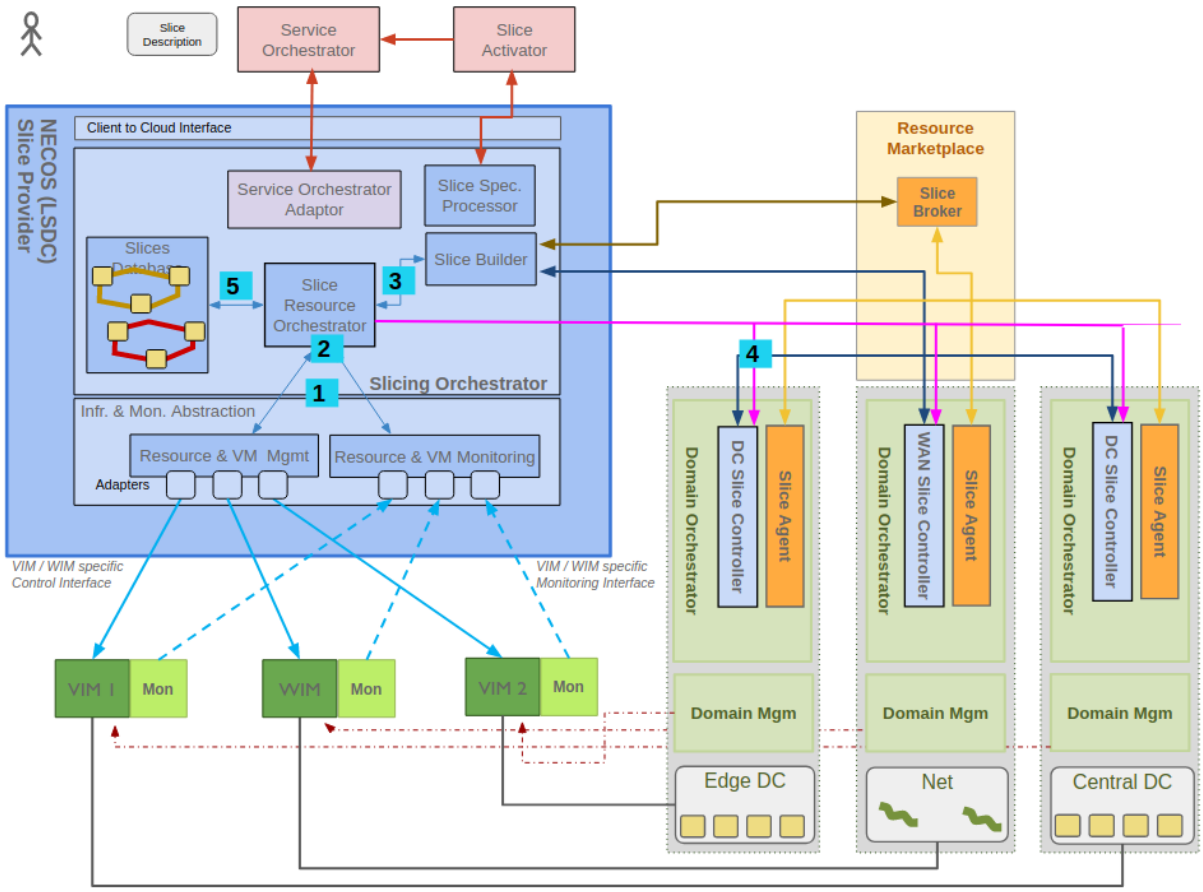

Figure 53. Steps of run-time and decommissioning workflow

6. After adding or removing slice parts, the SRO updates the associated slice topology in the Slice Database.

6.3 Cross-domain spanning

In this section we consider how the modes manifest with respect to the architecture, and present the outcome of a slice spanning across multiple domains.

Each mode is shown together with its cross-domain aspect:

- Mode 0 - the VIM on-demand model with DC slicing that allows direct inter-domain orchestrator to VIM interaction using a specific allocated VIM
- Mode 1 - the slicing in the VIM model, that allows direct inter-domain orchestrator to VIM interaction using a specific allocated shim object

Figure 41. Slices overlaying multiple domains using Mode 0shows how a slice overlays the architectural elements in a mode 0 manifestation. In mode 0 there is no Inter-domain Orchestrator interaction, but there is a VIM on-demand allocated for each slice part. This allows a direct Inter-domain Orchestrator to VIM interaction.

Figure 42 shows how a slice overlays the architectural elements in a mode 1 manifestation. In mode 1 there is also no Inter-domain Orchestrator interaction, however there is just One VIM at the remote cloud. To facilitate slicing, the remote cloud can allocate a small shim object that looks like a VIM from outside the domain, but is configured only for the remote client. The shim interacts with the one VIM to support the functionality needed.

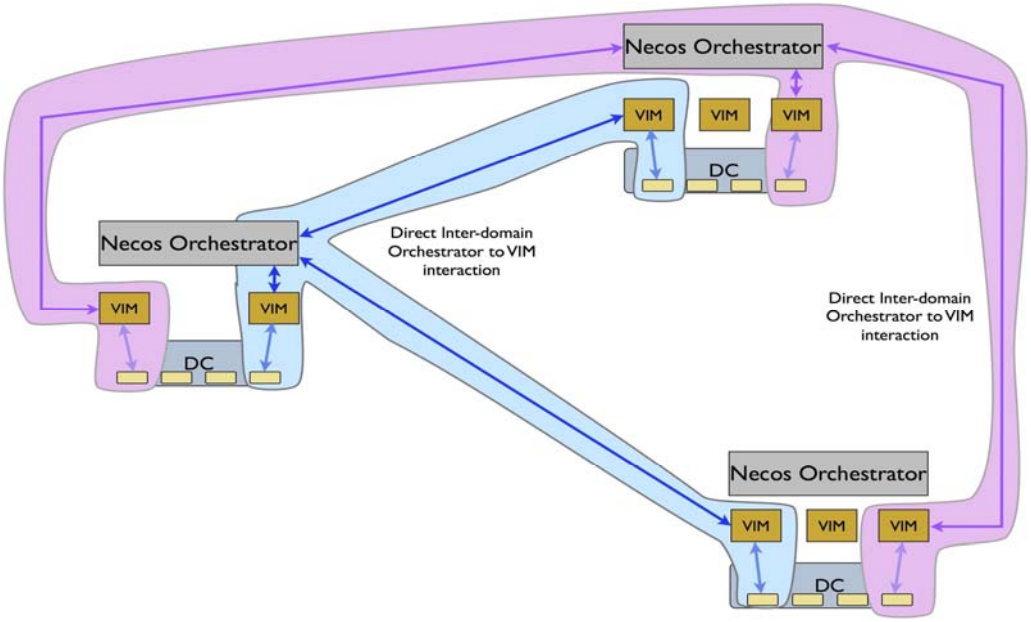

Figure 54. Slices overlaying multiple domains using Mode 0

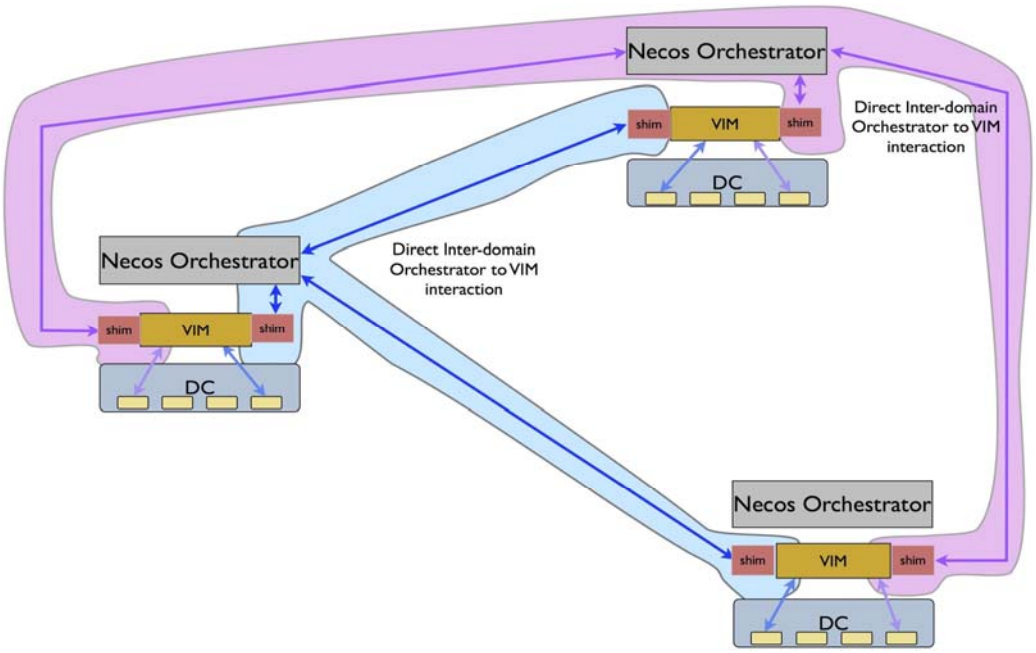

Figure 55. Slices overlaying multiple domains using Mode 1

7 Conclusions

The NECOS architecture described in Deliverable D3.1 is in support of *infrastructure slicing* to provide S*lice as a Service* as established in the project objectives. The Lightweight Slice Defined Cloud (LSDC) stands as the core element of this architecture, which is an approach for automating the configuration process of clouds and their interconnecting networks aimed to substantiate the infrastructure slicing. This is layered on the partitioning and virtualization of resources in data centers and wide area networks that are participating in the slice lifecycle, and also by providing a uniform management with a high-level of automaticity for the currently separated computing, connectivity, and storage resources.

The NECOS architecture provides some novel artefacts for providing the LSDC, and for network slicing in general. These are contained in the three main high-level sub-systems, which are (1) the NECOS (LSDC) Slice Provider, (2) the Resource Marketplace, and (3) the Resource Providers. These sub-systems are provided in order to support the tenants of NECOS who wish to use Slice as a Service. The NECOS Slice Provider (LSDC) is the sub-system that allows for the creation of full end-to-end Slices from a set of constituent Slice Parts. In NECOS, a Slice looks the same as the full set of federated resources, with the main attribute being that the domains look a lot smaller, as they have been sliced. The Resource Marketplace provides the way for the NECOS (LSDC) Slice Provider to find the slice parts to build up a slice. Rather than having a pre-determined set of providers that have been configured in a federation, we chose to use a more flexible model of a marketplace from which we could provision slice parts. The Resource Providers are organisations that can provide the resources required for the slice parts - namely, Data Center resources in the form of servers, storage, and network resources. Further resources can be provided by organisations that have Mobile Edge, Sensor Networks, Wireless, etc. Each provider will be capable of providing slice parts, which will be part of a full end-to-end slice.

The NECOS architecture has some interesting characteristics. It allows for slicing to be performed either at the physical infrastructure level or at the VIM/WIM management level with relatively small changes in the software components of one alternative in respect to the other. The tenant's orchestrator interacts directly with VIM/WIM elements created on-demand, or with shim objects of the virtualized VIM/WIM at each local domain. From the tenant's point of view, these alternatives offer more control on the resources. From the service providers point of view this low-level slicing approaches can be considered lightweight because they do not need to provide large slicing capable VIM/WIM or orchestrators in support of slicing, and the providers can participate directly in a slice marketplace.

Furthermore, the slice may be built assembling parts of resources that belong to different administrative domains, as the NECOS architecture follows a multi-domain approach. To participate in NECOS ecosystem, a given domain has to implement the appropriate APIs to offer resources through a marketplace. The NECOS LSDC contacts the marketplace so as to decide, based on different criteria, which resources will constitute the slice parts. Once this is done, the LSDC takes control of the resources at each local domain through on-demand created VIM/WIM or shim objects. Such a process is one of the main distinguishing features of NECOS in respect to other slicing architectures, which require a peer-to-peer interaction between the resource orchestrators of participating domains.

In the next version of this architecture (Deliverable D3.2), we plan to evolve the architecture and the design elements in more detail, and to consider the modes again. We plan to consider the implications of having virtual hosts as part of a slice. Although adding virtual hosts gives a level of flexibility and lightweightness, it also increases the complexity. This complexity also applies to other functionalities that will be revisited and evolved. In summary, the present architecture is the initial version of an evolving process that will considerer the lessons learnt in the subsequent evaluation process.

Further input and updates of the NECOS architecture will come from the use-cases being extended in WP2, and from experience of WP4, which provides the API specifications to request and provision slices, enabling *Slice as a Service*. The enhanced use-case requirements and the information model and the cloud APIs defined in WP4, will provide direct feedback, especially as the model and the APIs are evaluated during proof-of-concepts implementation and evaluation.

8 References

[IN1] https://cloudify.co/

[IN2] https://www.terraform.io/

[IN3] https://www.scalr.com/

[IN4] http://www.5gex.eu/

[IN5] http://www.fp7-unify.eu/

[SS1] "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions", I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini and H. Flinck, in IEEE Communications Surveys & Tutorials, vol. 20, no. 3, pp. 2429-2453, third quarter 2018.

[SC1] "Programmable Networks for IP Service Deployment", Galis, A., Denazis, S., Brou, C., Klein, C. – ISBN 1-58053-745-6, pp 450, June 2004, Artech House Books, Online: http://www.artechhouse.com/International/Books/Programmable-Networks-for-IP-Service-Deployment-1017.aspx

[SC2] "Management and Service-aware Networking Architectures (MANA) for Future Internet" – A. Galis *et al* - Invited paper IEEE 2009 Fourth International Conference on Communications and Networking in China (ChinaCom09) 26-28 August 2009, Xi'an, China, http://www.chinacom.org/2009/index.html

[SC3] "The RESERVOIR Model and Architecture for Open Federated Cloud Computing", Rochwerger, J. Caceres, R. Montero, D. Breitgand, A. Galis, E. Levy, I. Llorente, K. Nagin, Y. Wolfsthal; the IBM System Journal Special Edition on Internet Scale Data Centers, vol. 53, no. 4, 2009, http://www.haifa.ibm.com/dept/stt/sas_public.html

[SC4] "Infrastructure Slicing Landscape: –Galis. A, Makhijani, K - Tutorial at IEEE NetSoft 2018, Montreal 19 July 2018; http://discovery.ucl.ac.uk/10051374/

[SC5] http://groups.geni.net/geni/wiki/GENIConcepts

[SC6] ITU-T Y.3011- http://www.itu.int/rec/T-REC-Y.3001-201105-I

[SC7] https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-5G-Architecture-WP-July-2016.pdf

[SC8] https://5g-ppp.eu/wp-content/uploads/2018/01/5G-PPP-5G-Architecture-White-Paper-Jan-2018-v2.0.pdf

[SC9] "A Scientometric Analysis of Cloud Computing Literature"- Heilig, L., Voss, S. - IEEE Transactions on Cloud Computing, Volume: PP, Issue: 99, 30 April 2014, ISSN: 2168-7161; DOI: 10.1109/TCC.2014.2321168

[TR28801] "Telecommunication management; Study on management and orchestration of network slicing for next generation network", https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3091

Version History

Version	Date	Author	Change record
1.9	13/09/2018	Stuart Clayman	Complete draft to review
2.0	28/09/2018		Internal review
2.1	05/10/2018	Stuart Clayman	Final adjustments
2.2	04/03/2019	Stuart Clayman	Complete draft including project reviewers recommendations
2.3	12/03/2019	Stuart Clayman	Final updated version after internal review

Appendix 1 - Slicing in NECOS: VIM on-demand

The NECOS architecture will provide an automated provisioning and slicing of cloud resources which comply with customer requirements, as well as adapting them at run-time, in response to the system dynamics. Having Slice as a Service will be an advance in the state-of-the-art of current cloud computing architectures. This requires advances in heterogeneous resource management, federation and orchestration of compute, storage, and mobile entities in the form of slices that will enable an isolated, secure and scalable delivery of multi-cloud distributed services.

We have as a fundamental concept Data Center (DC) slicing, as part of the full NFVI foundation, to ensure that the attributes prescribed to network slices are propagated into the DC. We make the case for creating a VIM on-demand and dynamically allocating a new VIM (Virtual Infrastructure Manager) for each slice, rather than having one for the whole DC, which can be beneficial for those scenarios. There are some scenarios in which it is important to have a separate Data Center slice within a full Network Slice.

DC Slicing

In this section we present an overview of some of the mechanisms, components, and abstractions that can be utilized in order to encompass network slicing into a bigger picture for NFV delivery.

The DCs and the networks are physically connected. Slices can be requested from networks. This is on-going work in many arenas such as IETF and IEEE. Slices should also be a feature that can be requested from DCs.

DC Slice

A DC slice is an abstraction over the resources of a DC

- It presents a collection of resources that look like a DC
- It can be controlled and managed independently from any other slices

A DC slice can be allocated at any Data Center: large centralised DCs, medium DCs, and mobile edge DCs.

A slice is a basis for control in virtualized environments:

- a DC slice needs to be as elastic as other elements
- it can grow or shrink dynamically under the control of a Slice Controller

For each slice there will be an on-demand VIM allocated for any kind of lower level virtualization, including Xen, kvm; or for containers such as Docker / kubernetes. As a consequence, this choice is not a feature pre-determined once by the DC or the provider, but can now be an option for the customer. The customer will have a lot more configuration options for their VIM. It also means that the customer could also be billed for their VIM as opposed to it being part of the shared infrastructure.

VIM on-demand Deployment

A VIM needs to be allocated for each slice. We need a management component in each domain which can allocate a slice. A slice owner can manage, configure, and control their own VIM. The actual VIM that is deployed could be one of many: e.g. OpenStack, OpenNebula, OpenVIM, Kuberbetes. The VIM will be chosen from a pre-determined catalogue of VIMs that the DC owner has. As a VIM can be its own distributed system we need to decide how and where to deploy the VIM components.

Structural View

The following figures show the structural view of how a DC can be sliced. We see it from the viewpoint of a single Data Center. These DC slices are then composed into a single multi-domain topology to form a NECOS slice.

One VIM Per DC

This approach has one VIM for all of the resources of the DC. This is the current situation in DCs. The VIM gets more and more functionality in order to do more or different tasks.

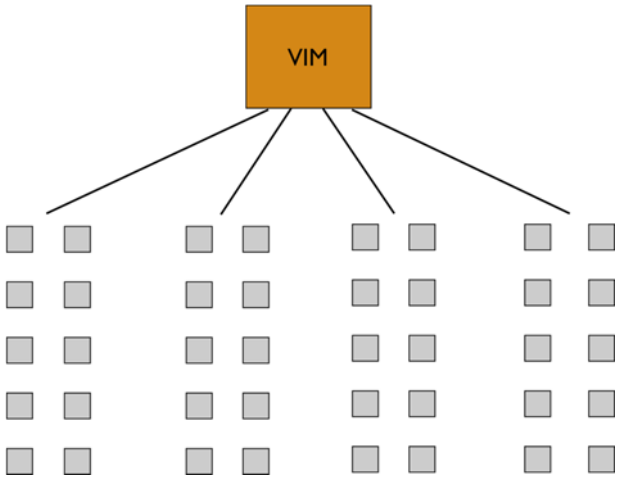

Figure A1.1. One VIM per data center

This approach has one VIM for all of the resources of the DC. This is the current situation in DCs. The VIM gets more and more functionality in order to do more or different tasks. The VMs for the NFVs are scattered across the infrastructure, and each slice is inter-mingled with the others. The attributes of the network part of the slice do not apply in the DC.

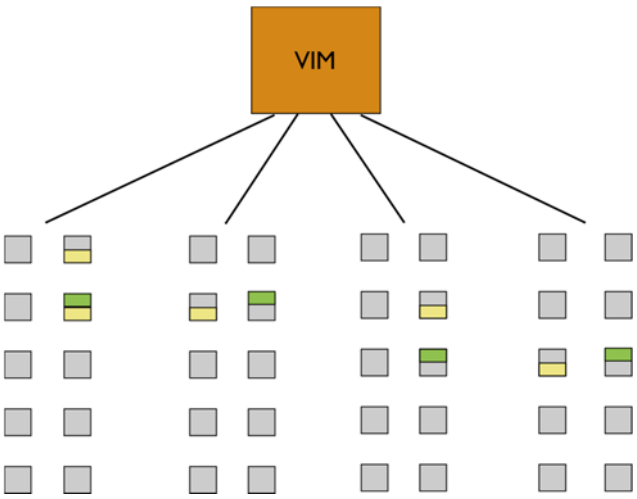

Figure A1.2. One VIM per DC with scattered NFVs

One VIM for all slices

The approach of one VIM for all of the resources of the DC has many issues when we introduce slices. It forces all of the slices to have the same strategies and policies, as there is only one VIM. The owner of each slice does not have the flexibility to control the slice how they wish, with different placement strategies of different approaches to energy management. The VIM needs to be even more complex to deal with this.

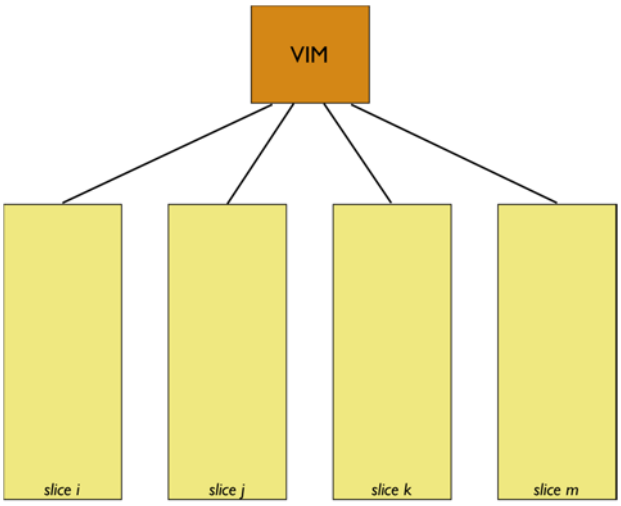

Figure A1.3. One VIM for all slices

We have found an approach that overcomes these complexities, and allows us to build modular and scalable slices.

One VIM per Slice

This approach has many VIMs - one per slice. Each VIM can have its own independent strategies and can be managed differently from the other slices. The slice owner can configure the own VIM as needed for their use. We need to ensure that each slice is isolated from the others, and that a VIM cannot manage the wrong resources. The one VIM per slice is a basic premise of our approach.

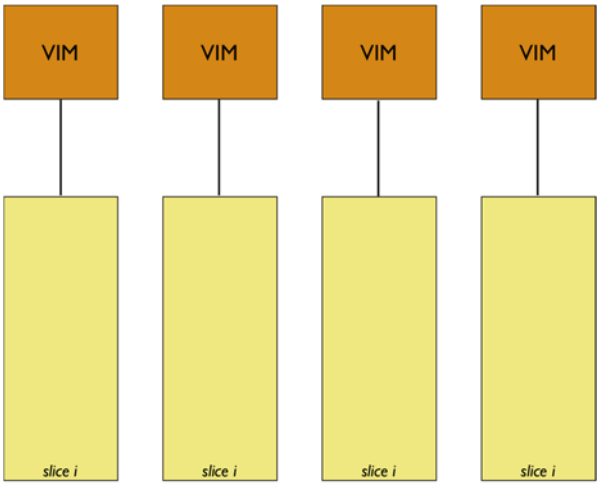

Figure A1.4. One VIM for each slice

A VIM is not special, and there is no requirement to have only 1, it is just another piece of software and can be started and stopped at any time.

Appendix 2 – Mapping of the requirements plus cross-WP feedback on the Architecture for D3.2

In this section, we present aspects related to the requirements derived from D2.1 and their impact to the design of the NECOS architecture. It is important that NECOS architecture conforms to the requirements presented in D2.1. Moreover, It is necessary to proceed a careful and continuous mutual evaluation for both requirements and architecture, given the fact that NECOS requirements elicitation and architecture development are ongoing efforts conducted in parallel. It demands a careful mapping between requirements and architecture components, as well as a thorough verification by a testing procedure.

Based on those assumptions, we have used the feature-oriented requirements modeling technique proposed by Liu and Mei[3]. According to them, "a functional feature can be mapped to a subsystem or a component. Non-functional features can generally not be pinpointed to a particular component, but have impacts on the system structure and behavior as a whole". Their approach can be divided into three main activities:

- Feature elicitation, focused on user requirements elicitation in terms of features;
- Feature analysis and organization, which organizes potential features into a tree hierarchy; and
- Feature refinement, elaborating a feature into a set of functionalities.

Feature elicitation was already done in the section Prioritisation of Requirements, given the project choice on focusing the requirements that are related to slice provisioning. Table following table presents the identified requirements:

5G Infrastructure (vRAN) Scenario	
RF.vRAN.3	*On-demand slice provisioning*
RN.vRAN.1	*Isolation of slice resources*
5G Services scenario, 4 Functional Requirements and 3 Non-Functional Requirements	
RF.5G.3	*On-demand slice provisioning*
RF.5G.4	*External control and management of the offered slices*
RN.5G.1	*Isolation of slice resources*
vCPE Scenario, 8 Functional Requirements and 5 Non-Functional Requirements	
RF.vCPE.1	*On-demand slice provisioning*
RF.vCPE.2	*Manageable slice*
RF.vCPE.3	*VIM-independence*
RF.vCPE.4	*Bare-metal slice*
RF.vCPE.6	*Elasticity*

[3] LIU, D., MEI, H., Mapping requirements to software architecture by feature-orientation, 2nd International Software Requirements to Architectures Workshop (STRAW 03), USA, 2003.

RN.vCPE.1	*Isolation of slice resources*
Content Delivery Touristic Services Scenario, 5 Functional Requirements and 5 Non-Functional Requirements	
RF.Touristic(CD).1	*Slice and slice-resource management*
RF.Touristic(CD).4	*Slice resource and service monitoring*
RN.Touristic(CD).3	*Elasticity*
RN.Touristic(CD).5	*Scalability*
Applications Touristic Services Scenario, 3 Functional Requirements and 4 Non-Functional Requirements	
RN.Touristic(APP).2	*Scalability*
RN.Touristic(APP).4	*Elasticity*
Emergency Scenario, 4 Functional Requirements and 3 Non-Functional Requirements.	
RF.emergency.1	*Dynamic slice management*
RF.emergency.3	*Timely slice management*

Features analysis and organization is done based on some criteria:

- The features to support a specific business process can be grouped and abstracted as a higher-level feature
- The features to support a specific user class can be grouped and abstracted as a higher-level feature
- A non-functional feature that is a constraint on a functional feature becomes a sub-feature of the functional feature.
- If a feature at user level is used to realize a feature at business level, then the former becomes a sub-feature of the latter.

Based on the requirements list presented above, we can identify several features sets:

- Slice Provisioning: RF.vRAN3, RF.5G.3, and RF.vCPE.1
- Isolation: RN.vRAN.1, RN.5G.1, and RN.vCPE.1
- Management: RF.5G.4, RF.vCPE.2, RF. Touristic(CD).1, RF.Emergency.1, RF.Emergency.3
- Elasticity: RF.vCPE.6, RN.Touristic(CD).3, RN.Touristic(APP).4
- Scalability: RN.Touristic(CD).5, RN.Touristic(APP).2
- Monitoring: RF.Touristic(CD).4
- VIM-independence: RF.vCPE.3
- Bare-metal slice: RF.vCPE.4

Here, we consider that scalability means "increasing" the capacity to meet the "increasing" workload, which presents this concept as a subset for elasticity, considered herein as "increasing or reducing" the capacity to meet the "increasing or reducing" workload. Moreover, “Bare-metal slice” and “VIM-independence” are taken as slice provisioning special characteristics. Thus, given the features analysis and organization criteria defined before, the following figure presents a requirements tree hierarchy:

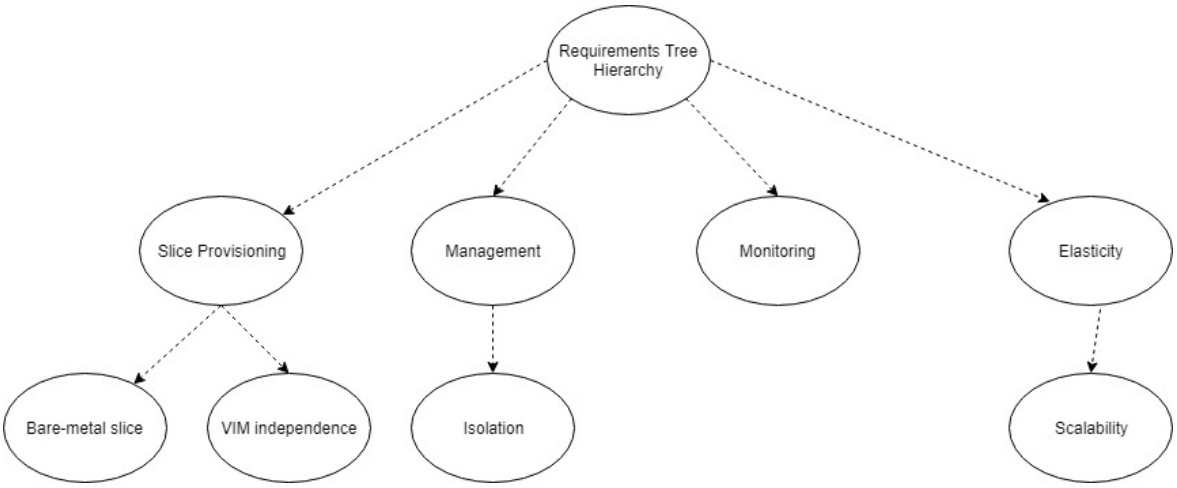

Figure A2.1. Requirements hierarchy tree

We can state that the most relevant features are present in NECOS, as described below:

- **Slice provisioning:** throughout the whole document, there are several functional architecture features related to the support for rapid resource/service provisioning.
  As previously described, to accomplish service provisioning tasks over the slices, NECOS presents a mechanism to support the slicing of the DC compute and storage resources, as well as the network resources. To do so, NECOS has two components at the infrastructure level that act as a point of control and management of DC (DC Slice Controller) and Network (WAN Slice Controller) slices.

  The mechanism to find these slice parts is described in the Resource Marketplace section. To participate in a Marketplace, a resource provider needs to run a Slice Agent, which can find available resources within the local resource domain, and then provide an offer for those resources to the Resource Broker (in the Marketplace), with a specified timespan and pricing model associated to it.
- **Management:** the tasks related to this feature are executed by LSDC Slice Provider. It contains the Slice Resource Orchestrator, which combines the slice Parts that make up a slice into a single aggregated slice. It is responsible for the orchestration of several elements, like:
  - the running end-to-end slices;
  - the service elements across the slice parts that make up the full end-to-end slice; and
  - the actual placement and embedding of VMs and virtual links for the services into the resource domains.
- **Monitoring:** related features are also executed by the NECOS (LSDC) Slice Provider, which has an Infrastructure and Monitoring Abstraction (IMA) mechanism. This allows the Slice Provider to interact with various remote VIMs, WIMs, and monitoring sub-systems in a generic way, using plug-in adaptors with the relevant API interactions. Besides that, it allows the Slice Resource Orchestrator to interact with the remote clouds in order to provision the actual tenant services, and to monitor the remote resources running those services.
- **Elasticity:** as presented in the section End-to-End Slice Elasticity, when a slice has to be augmented with resources regardless of their location (or other requirements that cannot be fulfilled with the already allocated Slice Parts), the Slice Resource Orchestrator (SRO) will contact the Slice Builder to delegate the process of looking for additional resources that can be attached to the existing slice as new slice parts.

The Slice Builder will in turn contact the Slice Broker in the marketplace, i.e., to discover resources in a similar manner as described for the slice instantiation phase. Once the handles to the new slice parts have been returned back from the Slice Builder, the SRO will perform the operations required to attach those Parts to the existing slice instance and will update the slice topology in the Slice Database.

In spite of its suitability, this effort is far from the end. There are several information model choices at the working packages under development (like D4.1) that will surely demand requirements and architectural revaluation. Thus, this section must be seen as an ongoing effort, describing a methodological approach to be adopted at NECOS final version documents.

As mentioned before, the NECOS architecture development is an ongoing activity and requires a careful mapping between its components and the respective requirements. A big challenge is building such an architecture that dynamically adapts itself to changing requirements.

With the usage of the methodology described here, we intend to facilitate the process of identifying new requirements, as well as, highlight the important requirements already proposed. It aims at generating a component architecture that supports this dynamic nature and which is easily inferred from a feature model. This scenario illustrates the need for updated descriptions in D3.2.

Appendix 3 – Requirements Analysis and Ranking

Quality Function Deployment - Introduction

Quality Function Deployment (developed by Y. Akao in Japan in 1966) is a systematic approach to design and develop a product (of any kind, including platforms/pieces of software) based on a close awareness of customer desires and requirements, coupled with the integration of functional groups (of a project team or company).

Quoting Y. Akao, QFD "*is a method for developing a design quality aimed at satisfying the consumer and then translating the consumer's demand into design targets and major quality assurance points to be used throughout the production phase. ... [QFD] is a way to assure the* ***design quality while the product is still in the design stage.***"

In essence, quality is the barycenter of the methodology and the ultimate goal is to translate (often) subjective quality criteria into objective ones that can be quantified and measured, and which can then be used to design and develop the product. Basically, QFD allows determining how and where priorities are to be assigned in product development apriori of implementation. The intent is to employ objective procedures in increasing detail throughout the development of the product or project.

The three main goals in implementing QFD are:

1. Prioritize spoken and unspoken customer desires and needs;
2. Translate these needs into technical characteristics, requirements and specifications;
3. Develop and deliver a quality product (or service) by focusing on customer/user satisfaction whilst optimizing usage of internal costs, resources and teams (e.g. in a project, or in a Company).

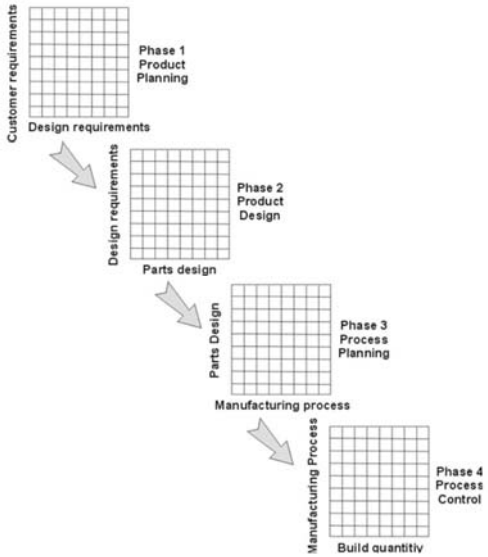

Figure A3.1. Flows of the QFD Analysis

QFD uses some principles from Concurrent Engineering in that cross-functional teams are involved in all phases of product development. As depicted in **Figure Appendix 3-1,** each of the four phases in a QFD process uses a matrix to translate customer requirements from initial planning stages through production control. These phases are:

- ***Phase 1: Product Planning***. It is also called The House of Quality. Main goals are: documenting customers' requirements, competitive opportunities, product measurements, competing product measures, and the technical ability of the organization to meet each customer requirement.
- ***Phase 2: Product Design***. Product concepts are created during this phase and part specifications

are documented. Parts that are determined to be most important to meeting customer needs are then deployed into process planning, or Phase 3.

- ***Phase 3: Process Planning***. During this phase, development processes are planned and flowcharted and the parameters (or target values) are documented.
- ***Phase 4: Process Control.*** It concerns the control of the development processes, the maintenance of schedules, and skills training for developers. Also, in this phase decisions are made as to which process poses the most risk and controls are put in place to prevent failures.

For each scenario in the NECOS project (i.e. 5G Networks Scenario, vCPE Scenario, Touristic Services Scenario, Emergency Service Scenario – reference Deliverable 2.1) we evaluated correlations between the requirements and NECOS Critical Success Factors/ NECOS Performance Indicators/ NECOS Expected Differentiated Factors (Characteristics). In other words we evaluated how each requirements is contributing to solve/enable each Project Critical Success Factors/ Project Performance Indicators/ Project Expected Differentiated Characteristics as seen from each scenario.

The Critical Success Factors/ Performance Indicators/ Expected Differentiated Characteristics used in the QFD analysis are as follows:

- ***Critical Success Factors (CSF):*** "CSF1 - Isolation-isolation is the factor that distinguish slicing from other cloud-based solutions. Since the slices are isolated from each other in all network, computing, and storage planes, the user experience of the slice will be the same as if it was a physically separate infrastructure; "CSF2 - Cost Reducing Any virtualization solution, being sliced or not, must benefit from its economy of scale, a multi-domain solution such as NECOS has to take advantage of the benefits of the market economy reaching scale through multiple providers, potentially specialized in their own virtualization domain (i.e., virtual networks, virtual computing, etc.) and therefore reducing its own costs and at the end, the costs transferred to the users; "CSF3 - Reliability The redundancy, geographic distribution, and technology diversity provided by a service that orchestrates different connectivity and computing services from different providers is a key factor for its reliability. As long as the orchestrator has the ability to quickly re-provision a slice after a failure is detected the reliability of the overall slice service may be higher than the reliability offered by each provider. ""CSF4 - Flexibility of cloud-based services should be one of its main distinguishing features. Much more on a multi-domain, diverse platform, as far as it works efficiently and transparently for the user. It is also closely related with reducing the costs for the user, as the resources used by the slice, and therefore its cost, can be reduced or augmented following the demands of the service deployed on top of it, avoiding overprovisioning. Additionally, the tenant must consider the flexibility of a lightweight solution such as NECOS, which intends to offer a basic service inside of which the tenant deploys its own services following its own policies and requirements; "CSF5 - Scalability - A particular case of such flexibility is the scalability of a provisioned slice. There are several definitions of scalability and elasticity, the CSF described below. For the scope of this project we define scalability as the ability to increase workload size within existing infrastructure (hardware, software, etc.) without impacting performance. We can think in scalability in two different dimensions: scalability of a particular provisioned slice and scalability of the number and size of the slices provided. "; "CSF26 - Elasticity As with scalability, there are more than one definition for elasticity. In the context of this project, elasticity is the ability to grow or shrink slice resources dynamically as needed to adapt to workload changes in an autonomic manner, maximizing the use of resources. "; "CSF7- Security Being a standard critical feature for any system, the security must be of particular attention for a solution based on sharing resources, isolation, and avoiding side channels must be a priority. "; "CSF8 - Slice Efficiency The critical factors mentioned above are no relevant if the communications and computing services provided by a slicing solution do not reach the performance expected and paid by the tenants/users. “; "CSF9 - Lifecycle Efficiency Additionally, a slicing service must be efficient regarding the own slice lifecycle: provisioning, monitoring, recovering from failures, etc. This efficiency is closely related with most of the factors mentioned above, from it depend factors such as cost, flexibility, and reliability. "; "CSF10 - Simplicity is obviously a factor of importance from the point of view

of the client but also an aspect with impact on other CSFs such as cost efficiency, but also on reliability and security. Understanding clearly what is being configured and provisioned is the first step to implement a service inexpensive, reliable and secure.

- ***Key Performance Indicators (KPI):*** KPI1-Average elasticity response time (in seconds); KPI2-Average end-to-end delay (in milliseconds, measured as half average RTT); KPI3-Average service provisioning time (in seconds); KPI4-Average slice provisioning time (in seconds); KPI5-Average throughput (in Mbps); KPI6-End-to-end slice availability (% of time); KPI7-Monitoring-data availability (%); KPI8 -Number of application users; KPI9-Physical Server utilization; KPI10-Service demand prediction accuracy; KPI11-Service disruption index; KPI12-Service QoE; KPI13-SLA fulfilment index; KPI14-Average Slice Decommission time (in seconds); KPI15-Slice isolation index; KPI16-Average Slice Provisioning time (in seconds).
- ***Expected Differentiated Factors (Characteristics) (DF):*** DF1 - LSDC empowers a new service model – the Slice as a Service, by dynamically mapping service components to a slice. The enhanced management for the infrastructure creates slices on demand and slice management takes over the control of all the service components, virtualized network functions and system programmability functions assigned to the slice, and (re) configure them as appropriate to provide the end-to-end service. DF2 - LSDC enables easy reconfiguration and adaptation of logical resources in a cloud networking infrastructure, to better accommodate the QoS demand of the Slice, through using software that can describe and manage various aspects that comprise the cloud environment; DF3 - LSDC allows each aspect of the cloud environment – from the networking between virtual machines to the SLAs of the hosted applications – to be managed via software. This reduces the complexity related to configuring and operating the infrastructure, which in turn eases the management of the cloud infrastructure; DF4- LSDC platform will offer the ability to a specific cloud provider to federate his own infrastructure with other cloud providers with different configurations in order to realize virtualized services through the use of the Slice as a Service concept. The users of the LSDC APIs and platform will be able to create virtual services that can span the merged cloud infrastructure offered by different cloud providers. This concept is not purely technical, it can also encompass business, cultural, geographical or in any other domain.

The scenarios identified in deliverable D2.1 and used in the QFD analysis are as follows:

- **Scenarios:** 5G Networks Scenario, vCPE Scenario, Touristic Services Scenario, Emergency Services Scenario

Correlation scores and importance factors for each problem were provided by an expert team of 17 members from NECOS partners. These correlation evaluations were done according to each scenario Quality context for each business objectives and project outcomes and also for all actors (i.e. *Overall value*: correlation values related to the average of each previous value).

Requirements Ranking Results of the QFD Analysis

The results of this QFD analysis are in terms of ranked requirements as they contribute most/least to the realisation of project *Critical Success Fac*tors/ project *Key Performance Indica*tors/ project *Expected Differentiated Factors (Characteristics*). These results will be used as input to the detailed design and implementation of the NECOS system components.

5G Networks Scenario View Point On project Critical Success Factors

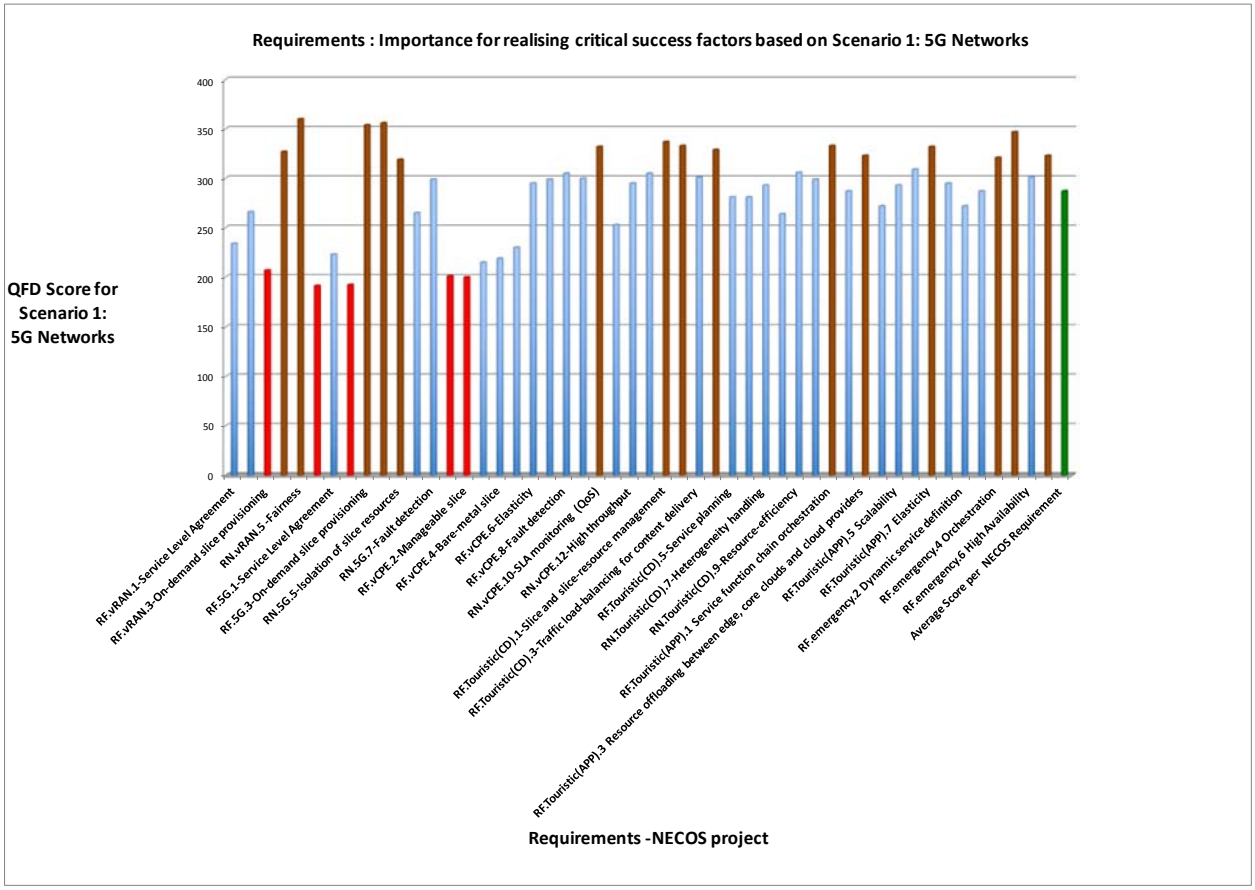

Figure A3.2. 5G Networks Scenario View Point– Realising Critical Success Factors

- 15 (of 50) requirements marked in brown colour are emerging as very important in realising Critical Success Factors as far as 5G networks Scenario is concerned.
- 5 (of 50) requirements marked in red colour are emerging as having a lower than average importance in realising Critical Success Factors as far as 5G networks Scenario is concerned.
- The rest of 30 (of 50) requirements marked in blue colour are emerging as having average importance in realising Critical Success Factors as far as 5G networks Scenario is concerned.

5G Networks Scenario View Point On project Key Performance Indicators

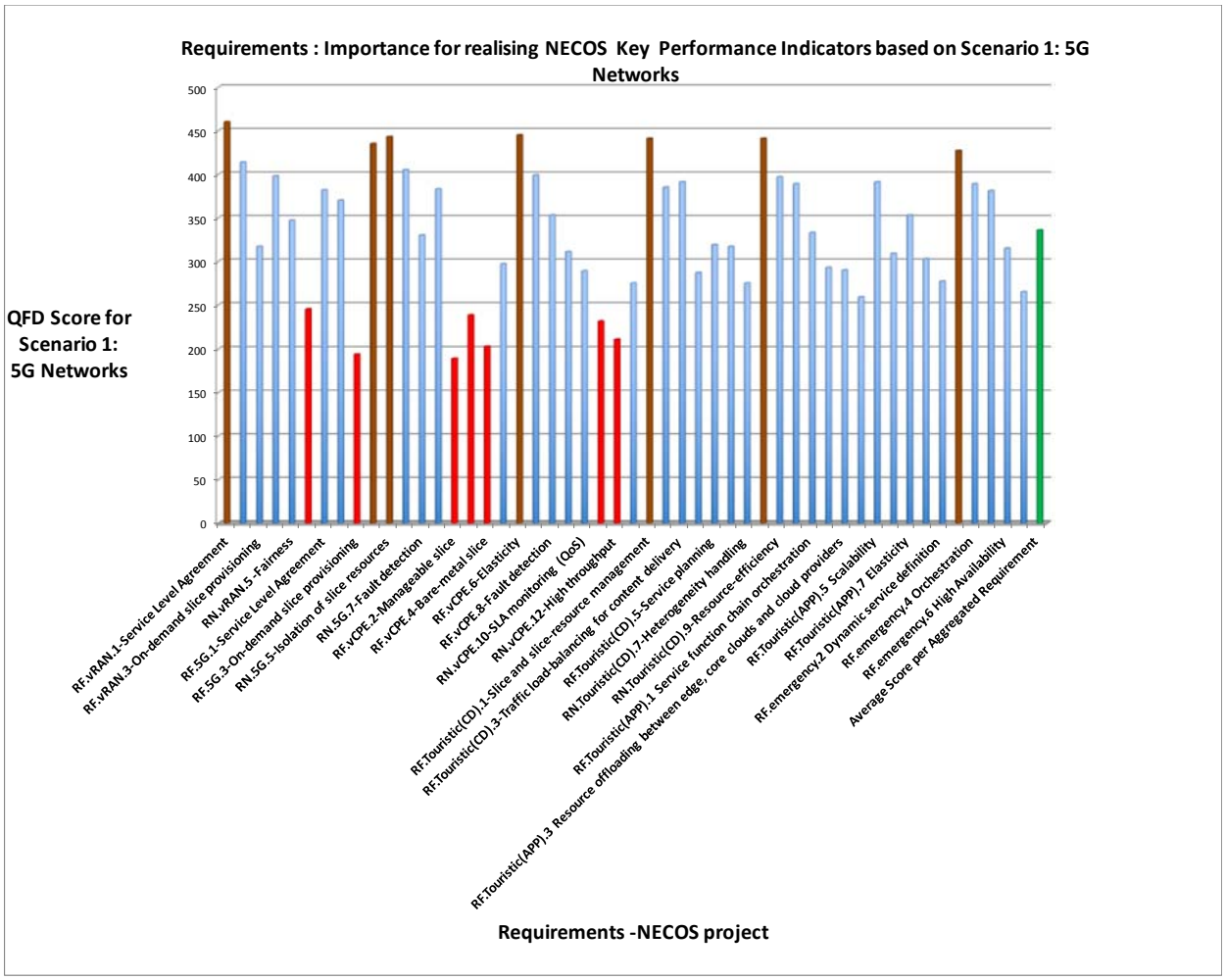

Figure A3.3. 5G Networks Scenario View Point– Realising Critical Success Factors

- 7 (of 50) requirements marked in brown colour are emerging as very important in realising Key Performance Indicators as far as 5G networks Scenario is concerned.
- 7 (of 50) requirements marked in red colour are emerging as having a lower than average importance in realising Key Performance Indicators as far as 5G networks Scenario is concerned.
- The rest of 36 (of 50) requirements marked in blue colour are emerging as having average importance in realising Key Performance Indicators as far as 5G networks Scenario is concerned.

5G Networks Scenario View Point On Project Expected Differentiated Factors (Characteristics)

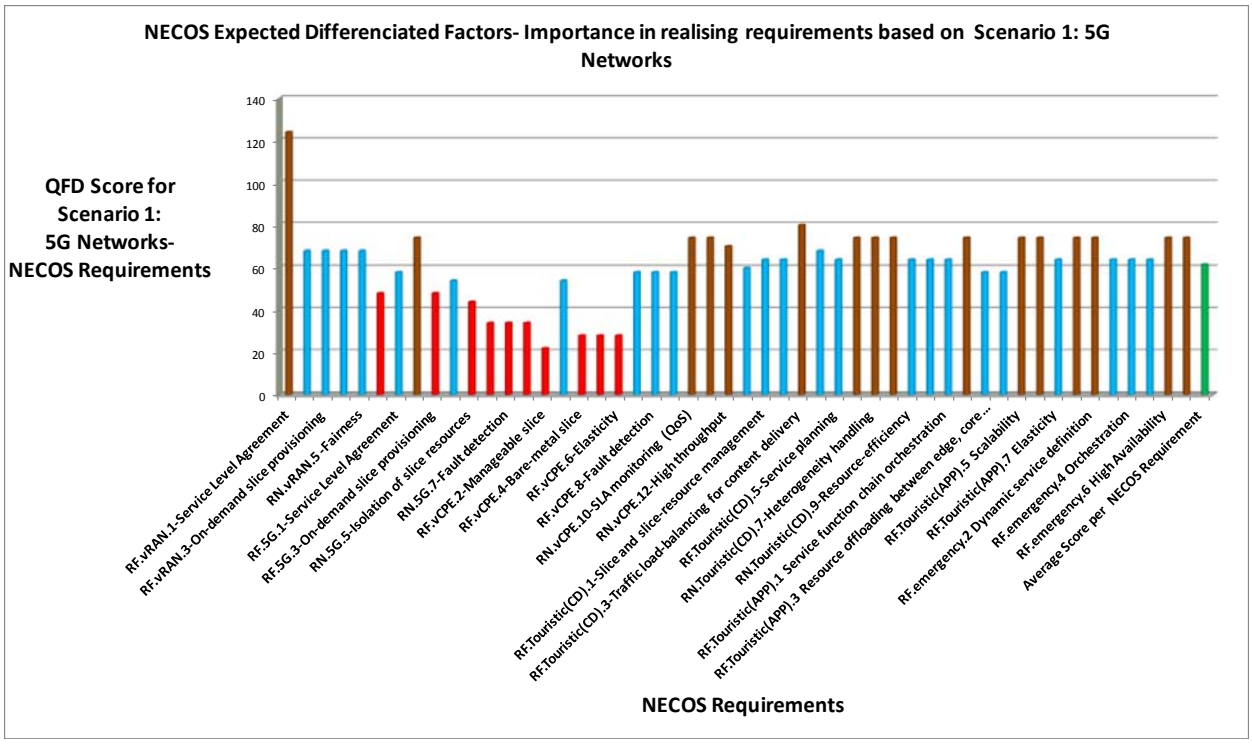

Figure A3.4. 5G networks Scenario View Point – Realising Expected Differentiated Factors

- 16 (of 50) requirements marked in brown colour are emerging as very important in realising Expected Differentiated Factors as far as 5G networks Scenario is concerned.
- 10 (of 50) requirements marked in red colour are emerging as having a lower than average importance in realising Expected Differentiated Factors as far as 5G networks Scenario is concerned.
- The rest of 34 (of 50) requirements marked in blue colour are emerging as having average importance in realising Expected Differentiated Factors as far as 5G networks Scenario is concerned.

vCPE Scenario View Point On project Critical Success Factors

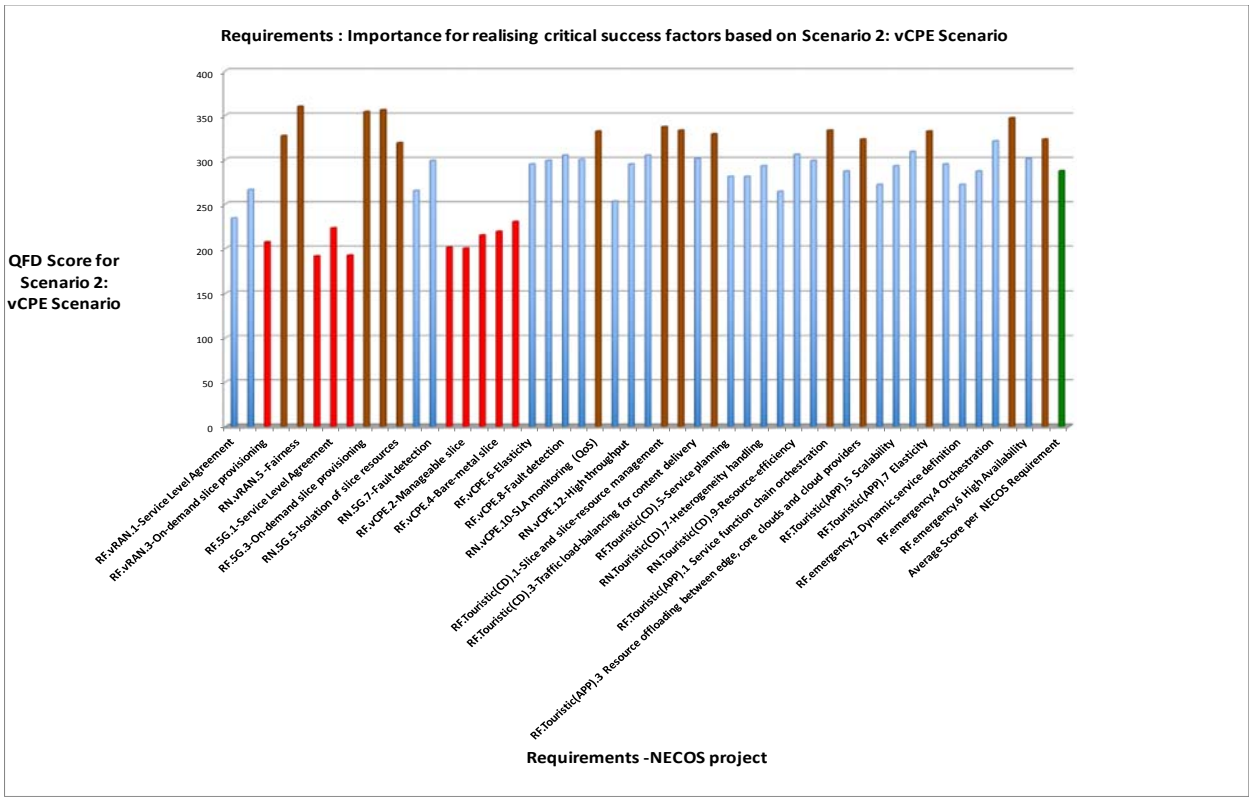

Figure A3.5. vCPE Scenario View Point– Realising Critical Success Factors

- 14 (of 50) requirements marked in brown colour are emerging as very important in realising Critical Success Factors as far as vCPE Scenario is concerned.
- 9 (of 50) requirements marked in red colour are emerging as having a lower than average importance in realising Critical Success Factors as far as vCPE Scenario is concerned.
- The rest of 27 (of 50) requirements marked in blue colour are emerging as having average importance in realising Critical Success Factors as far as vCPE Scenario is concerned.

vCPE Scenario View Point On project Key Performance Indicators

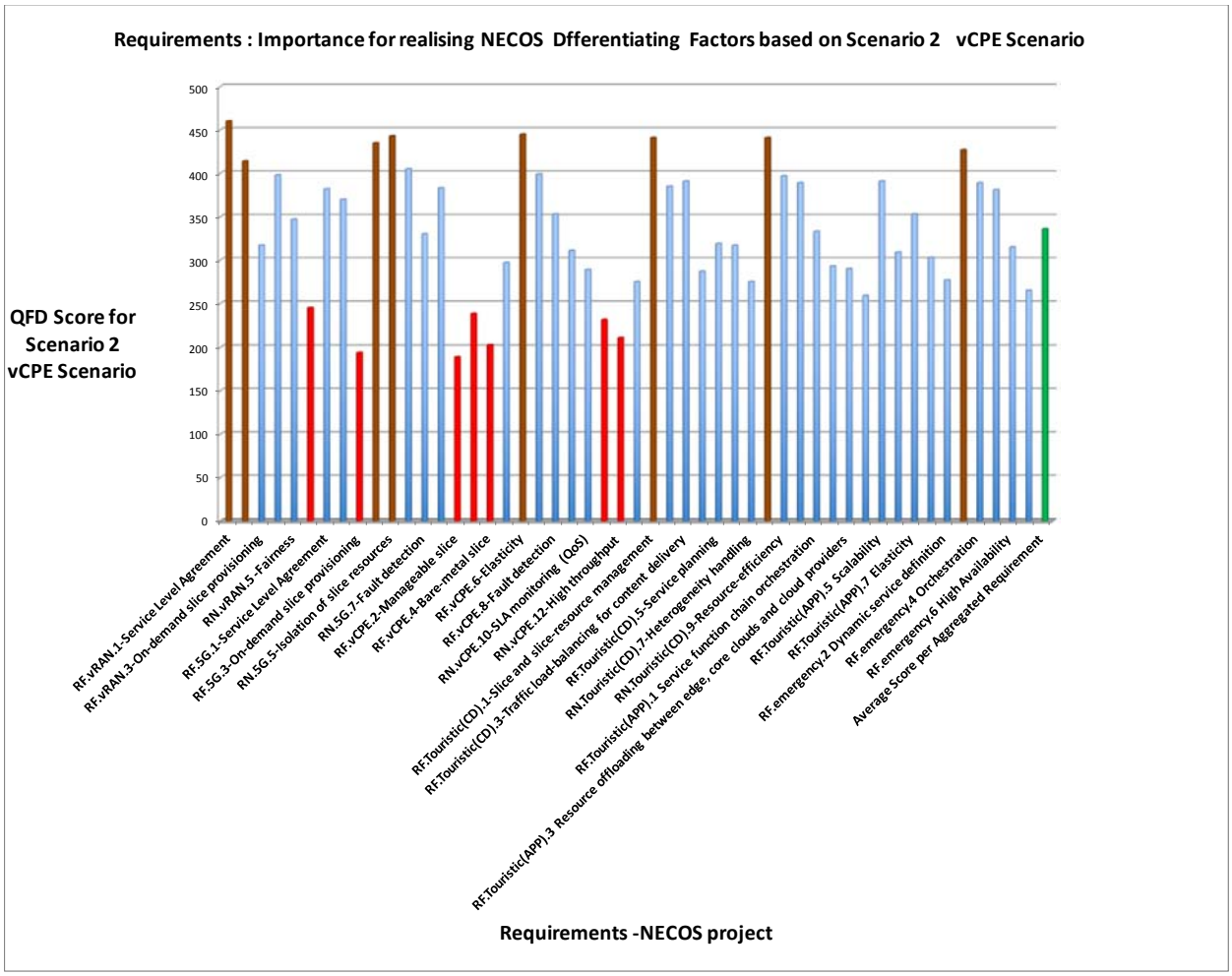

Figure A3.6. vCPE Scenario View Point– Realising Critical Success Factors

- 8 (of 50) requirements marked in brown colour are emerging as very important in realising Key Performance Indicators as far as vCPE Scenario is concerned.
- 7 (of 50) requirements marked in red colour are emerging as having a lower than average importance in realising Key Performance Indicators as far as vCPE Scenario is concerned.
- The rest of 35 (of 50) requirements marked in blue colour are emerging as having average importance in realising Key Performance Indicators as far as vCPE Scenario is concerned.

vCPE Scenario View Point On Project Expected Differentiated Factors (Characteristics)

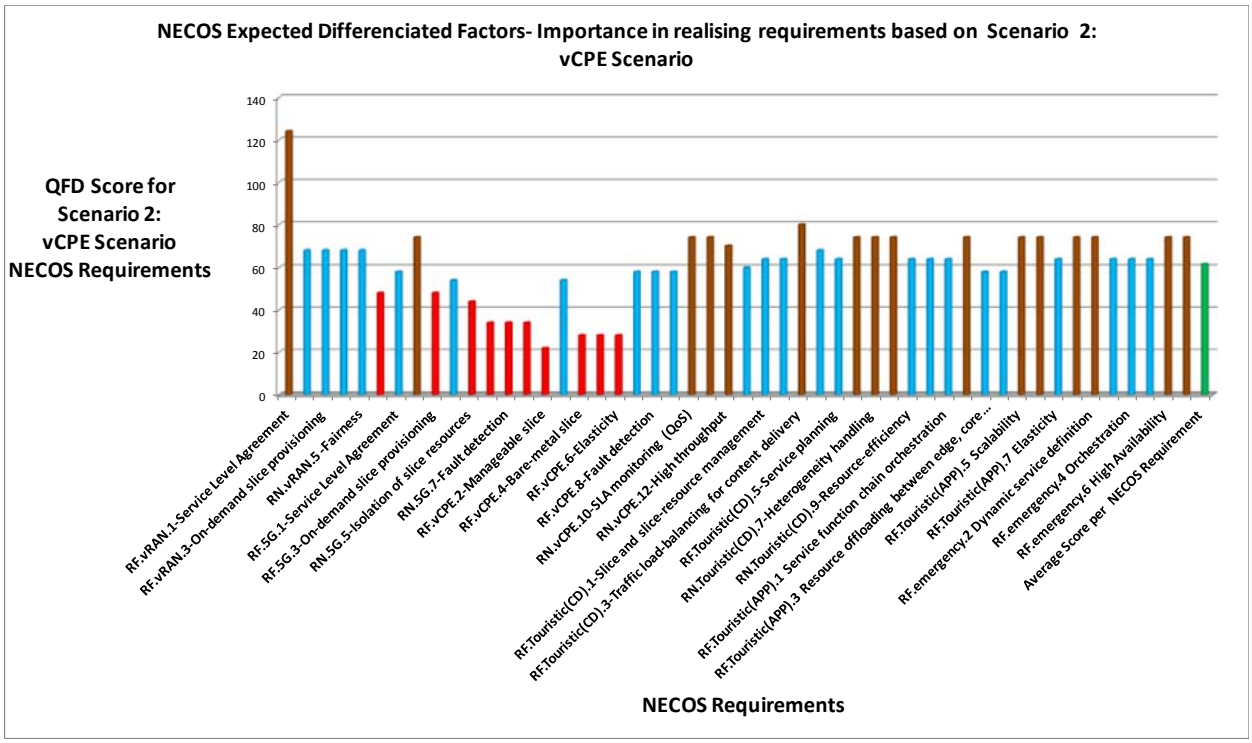

Figure A3.7. vCPE Scenario View Point – Realising Expected Differentiated Factors

- 16 (of 50) requirements marked in brown colour are emerging as very important in realising Expected Differentiated Factors as far as vCPE Scenario is concerned.
- 10 (of 50) requirements marked in red colour are emerging as having a lower than average importance in realising Expected Differentiated Factors as far as vCPE Scenario is concerned.
- The rest of 34 (of 50) requirements marked in blue colour are emerging as having average importance in realising Expected Differentiated Factors as far as vCPE Scenario is concerned.

Touristic Scenario View Point On project Critical Success Factors

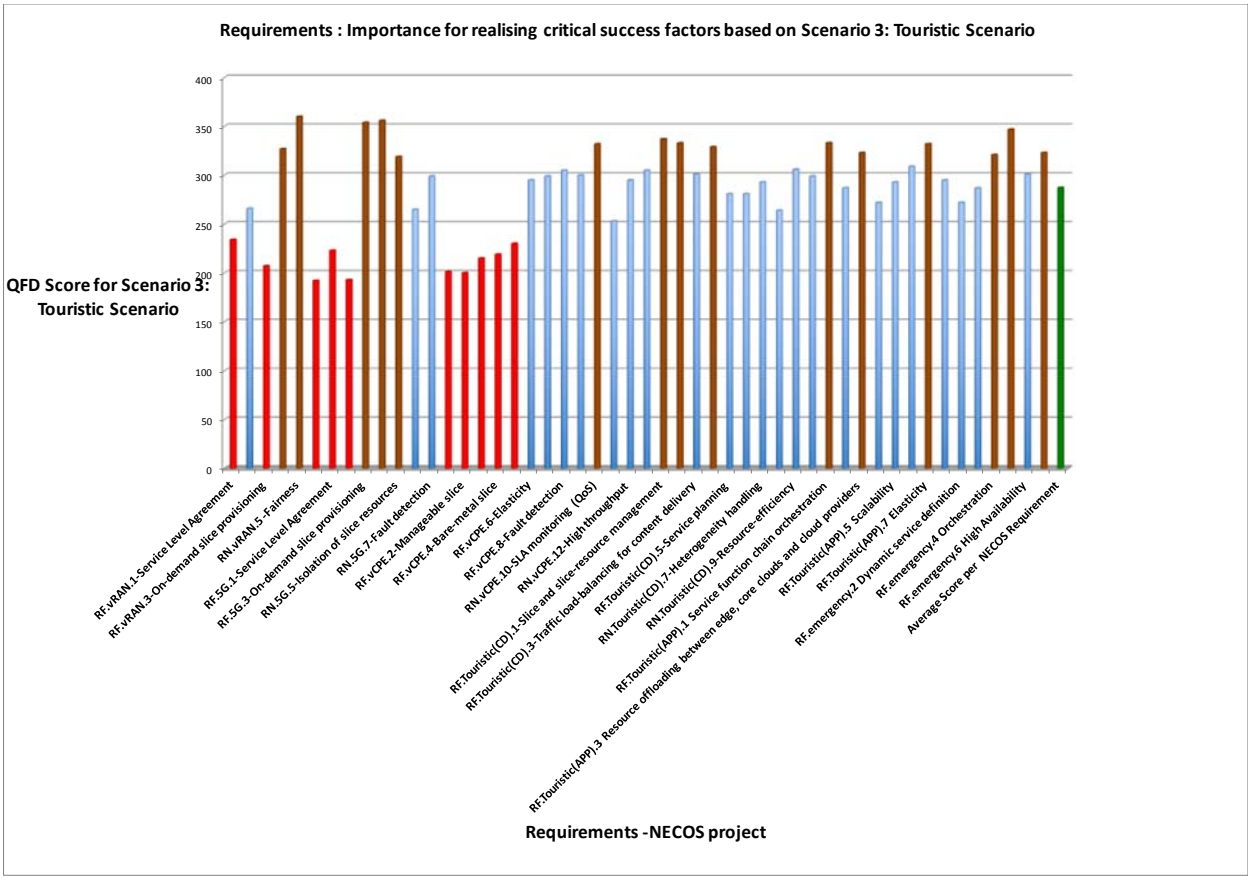

Figure A3.8. Touristic Scenario View Point– Realising Critical Success Factors

- 15 (of 50) requirements marked in brown colour are emerging as very important in realising Critical Success Factors as far as Touristic Scenario is concerned.
- 10 (of 50) requirements marked in red colour are emerging as having a lower than average importance in realising Critical Success Factors as far as Touristic Scenario is concerned.
- The rest of 25 (of 50) requirements marked in blue colour are emerging as having average importance in realising Critical Success Factors as far as Touristic Scenario is concerned.

Touristic Scenario View Point On project Key Performance Indicators

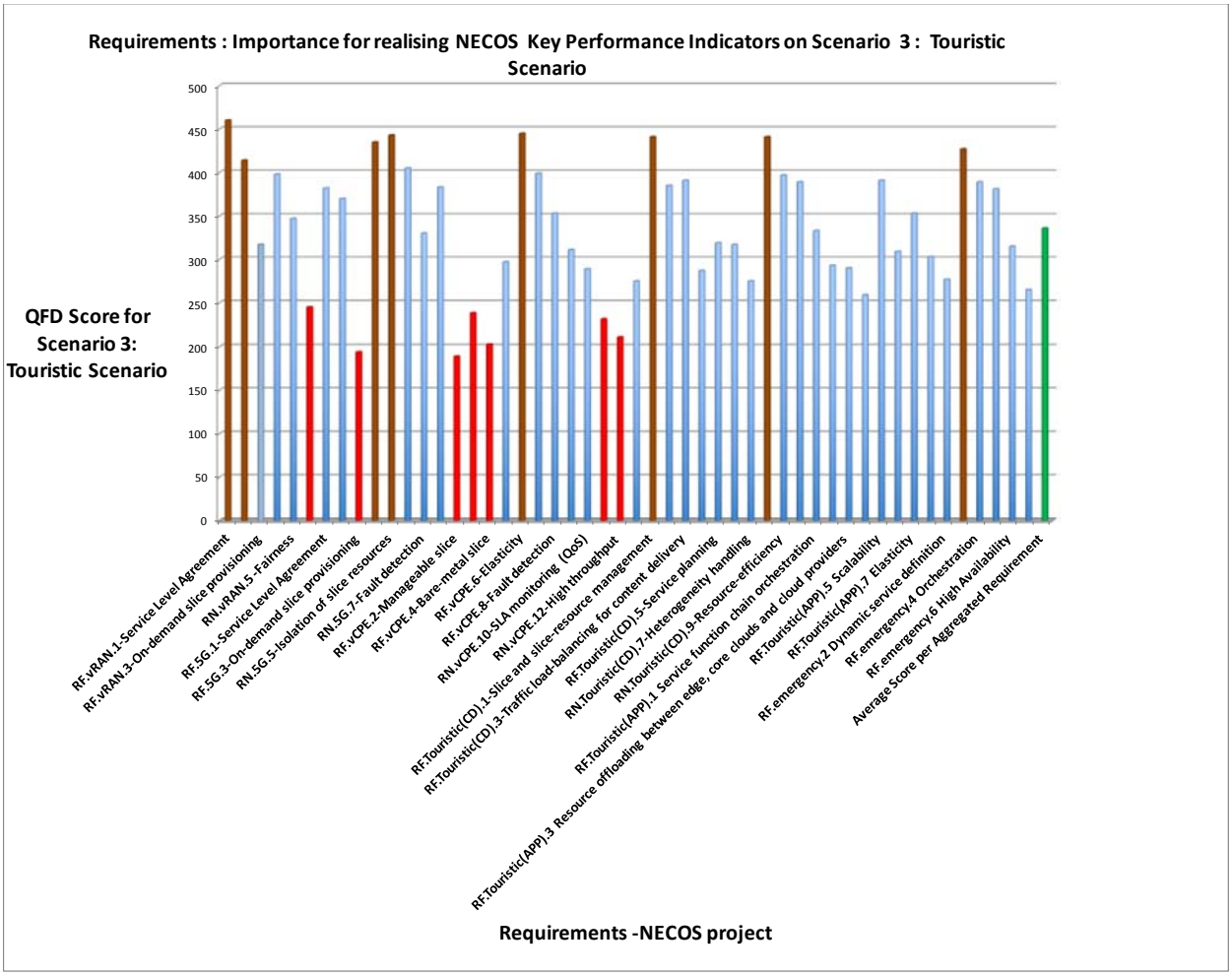

Figure A3.9. Touristic Scenario View Point– Realising Critical Success Factors

- 8 (of 50) requirements marked in brown colour are emerging as very important in realising Key Performance Indicators as far as Touristic Scenario is concerned.
- 7 (of 50) requirements marked in red colour are emerging as having a lower than average importance in realising Key Performance Indicators as far as Touristic Scenario is concerned.
- The rest of 35 (of 50) requirements marked in blue colour are emerging as having average importance in realising Key Performance Indicators as far as Touristic Scenario is concerned.

Touristic Scenario View Point On Project Expected Differentiated Factors (Characteristics)

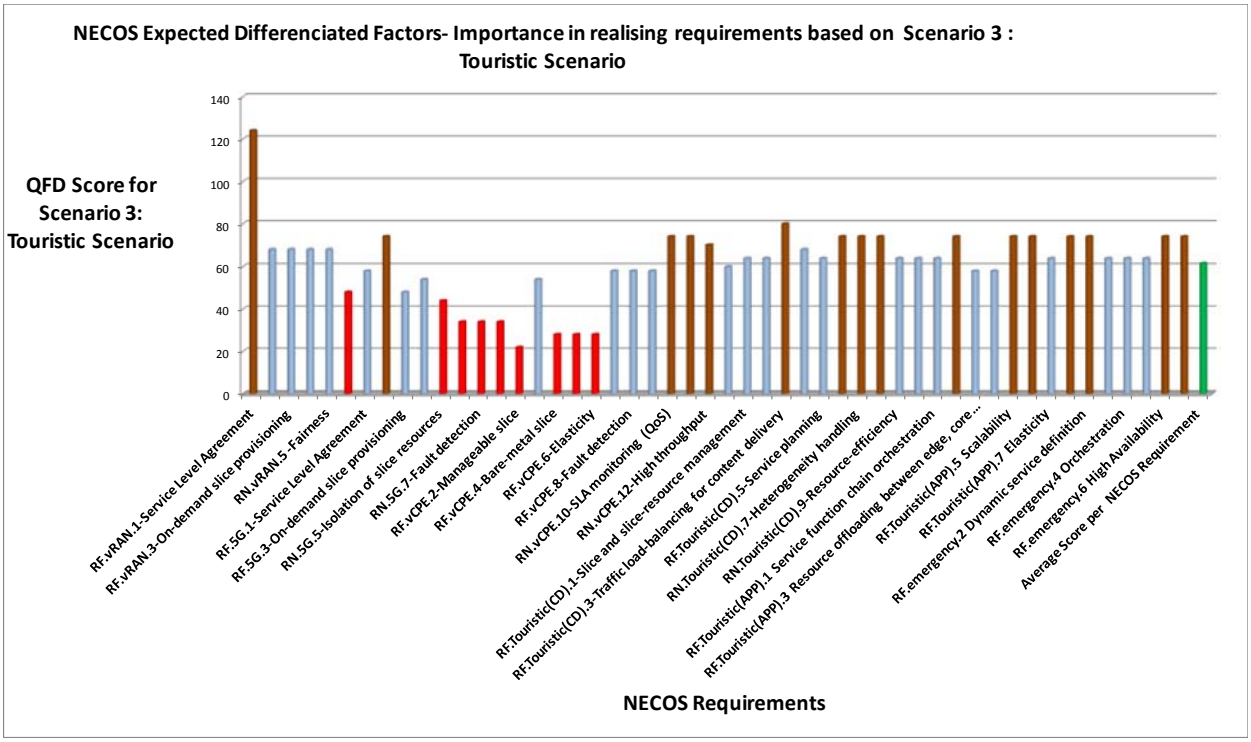

Figure A3.10. Touristic Scenario View Point – Realising Expected Differentiated Factors

- 16 (of 50) requirements marked in brown colour are emerging as very important in realising Expected Differentiated Factors as far as Touristic Scenario is concerned.
- 10 (of 50) requirements marked in red colour are emerging as having a lower than average importance in realising Expected Differentiated Factors as far as Touristic Scenario is concerned.
- The rest of 34 (of 50) requirements marked in blue colour are emerging as having average importance in realising Expected Differentiated Factors as far as Touristic Scenario is concerned.

Emergency Scenario View Point On project Critical Success Factors

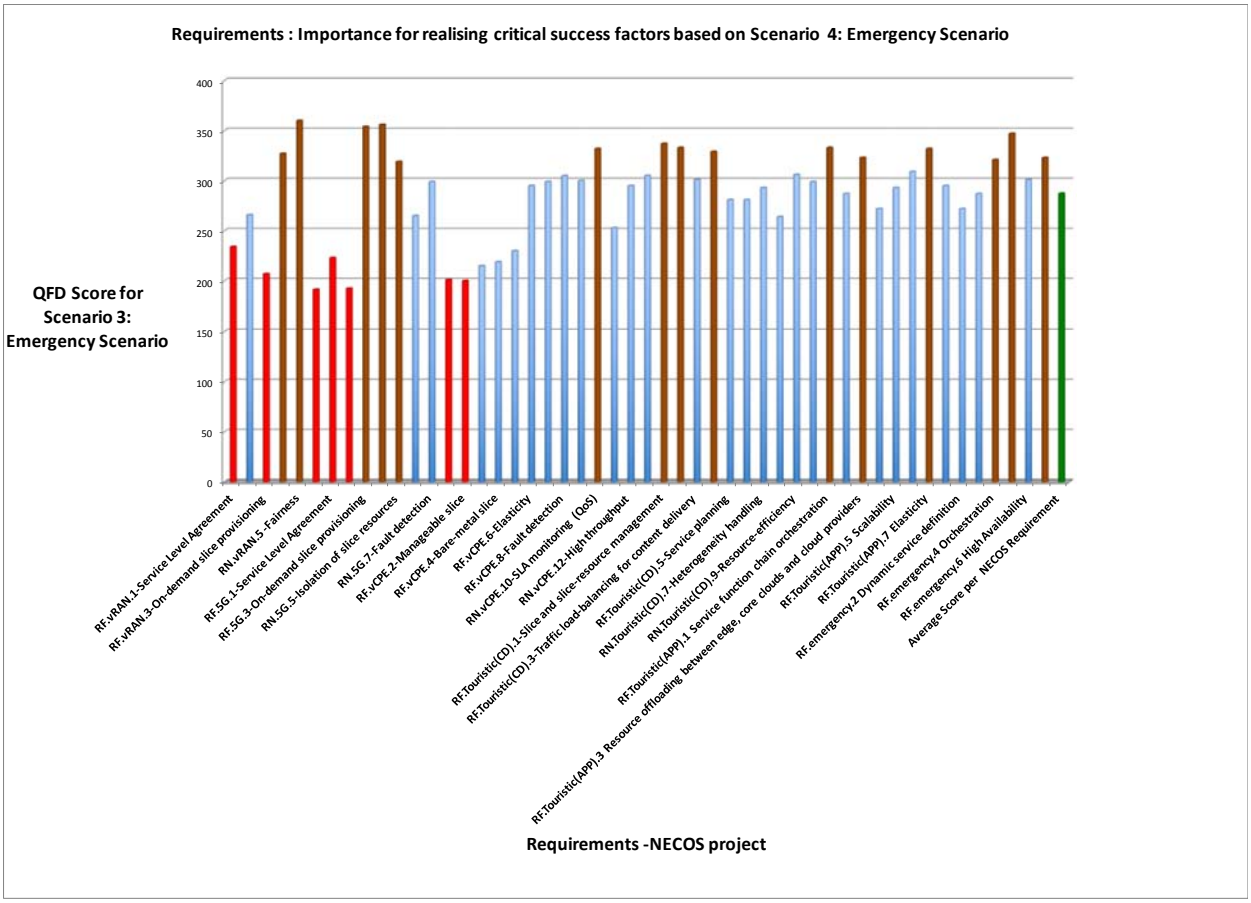

Figure A3.11. Emergency Scenario View Point– Realising Critical Success Factors

- 15 (of 50) requirements marked in brown colour are emerging as very important in realising Critical Success Factors as far as Emergency Scenario is concerned.
- 7 (of 50) requirements marked in red colour are emerging as having a lower than average importance in realising Critical Success Factors as far as Emergency Scenario is concerned.
- The rest of 38 (of 50) requirements marked in blue colour are emerging as having average importance in realising Critical Success Factors as far as Emergency Scenario is concerned.

Emergency Scenario View Point On project Key Performance Indicators

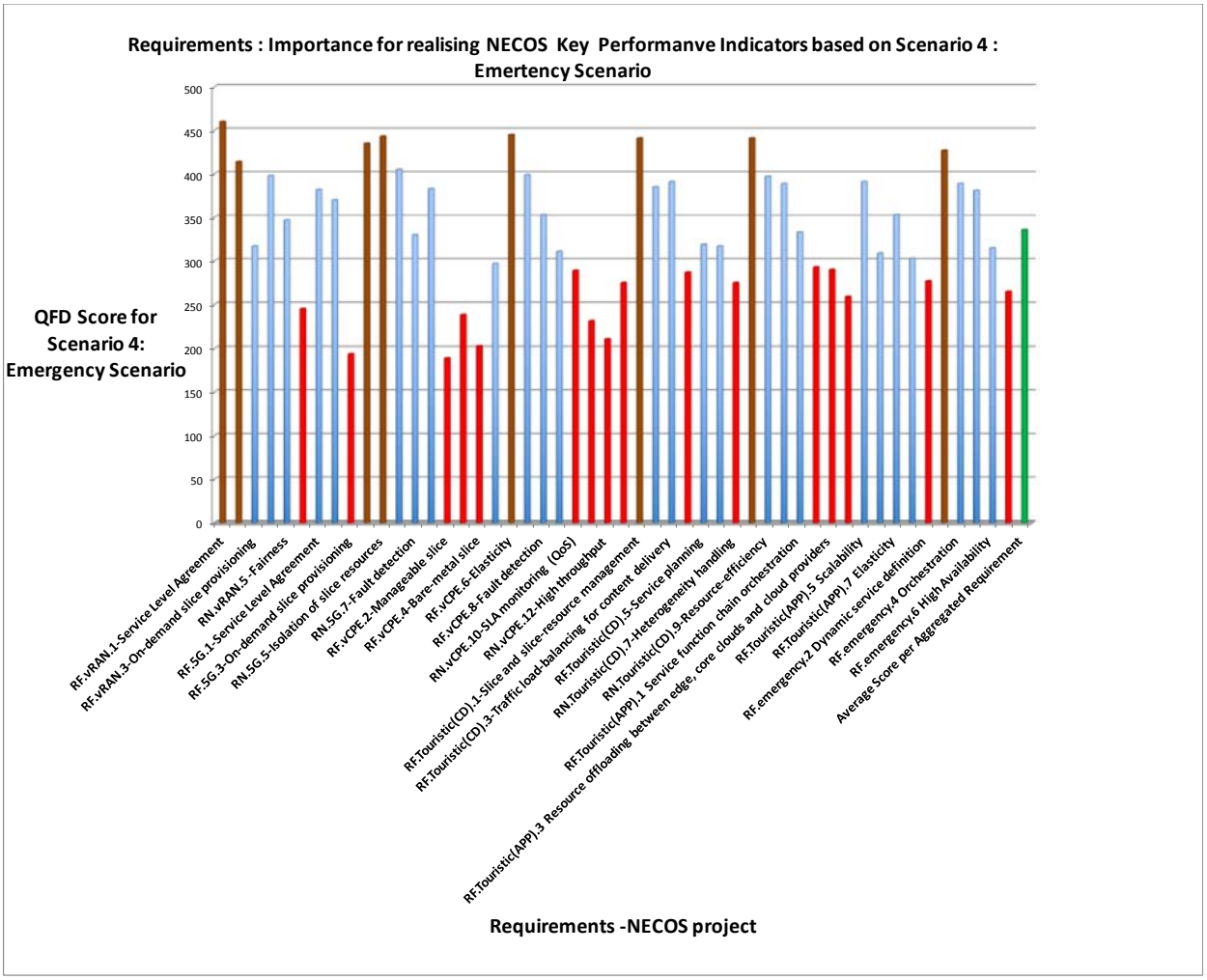

Figure A3.12. Emergency Scenario View Point– Realising Critical Success Factors

- 7 (of 50) requirements marked in brown colour are emerging as very important in realising Key Performance Indicators as far as Emergency Scenario is concerned.
- 7 (of 50) requirements marked in red colour are emerging as having a lower than average importance in realising Key Performance Indicators as far as Emergency Scenario is concerned.
- The rest of 36 (of 50) requirements marked in blue colour are emerging as having average importance in realising Key Performance Indicators as far as Emergency Scenario is concerned.

Emergency Scenario View Point On Project Expected Differentiated Factors (Characteristics)

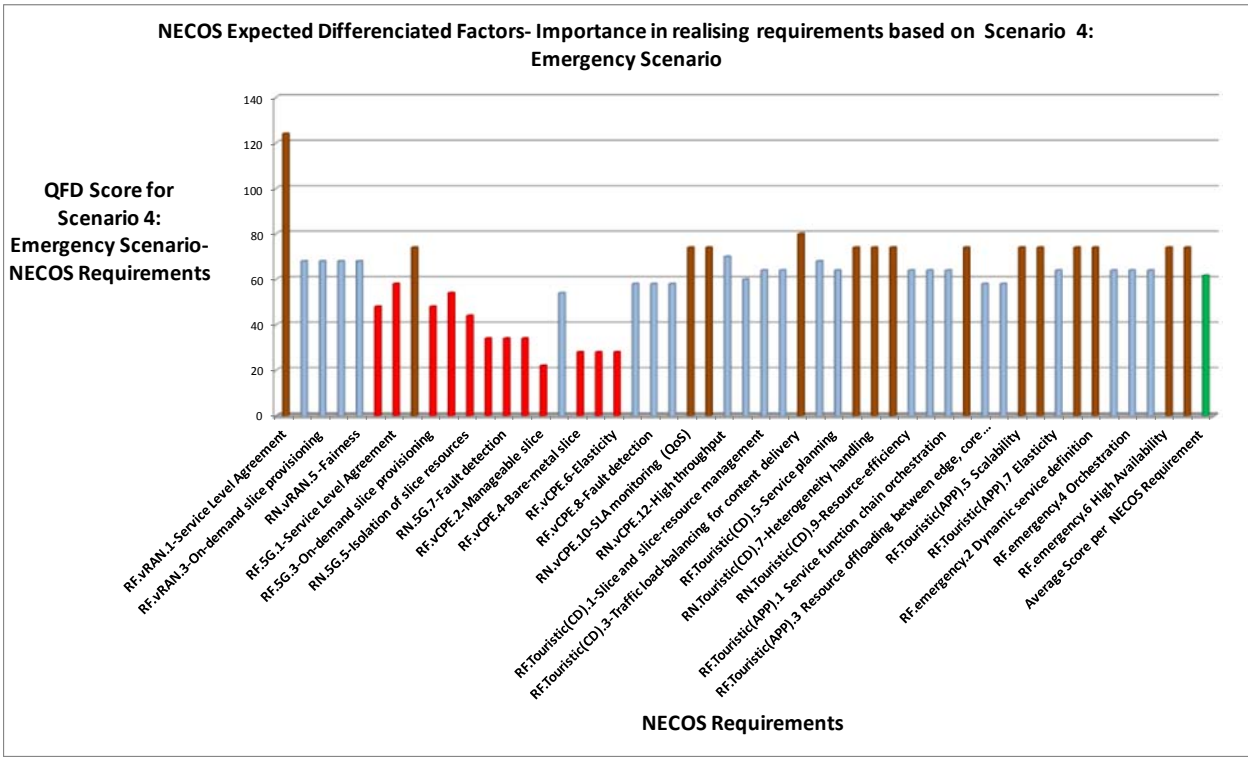

Figure A3.13. Emergency Scenario View Point – Realising Expected Differentiated Factors

- 15 (of 50) requirements marked in brown colour are emerging as very important in realising Expected Differentiated Factors as far as Emergency Scenario is concerned.
- 12 (of 50) requirements marked in red colour are emerging as having a lower than average importance in realising Expected Differentiated Factors as far as Emergency Scenario is concerned.
- The rest of 23 (of 50) requirements marked in blue colour are emerging as having average importance in realising Expected Differentiated Factors as far as Emergency Scenario is concerned.

All Scenario View Point On project Critical Success Factors

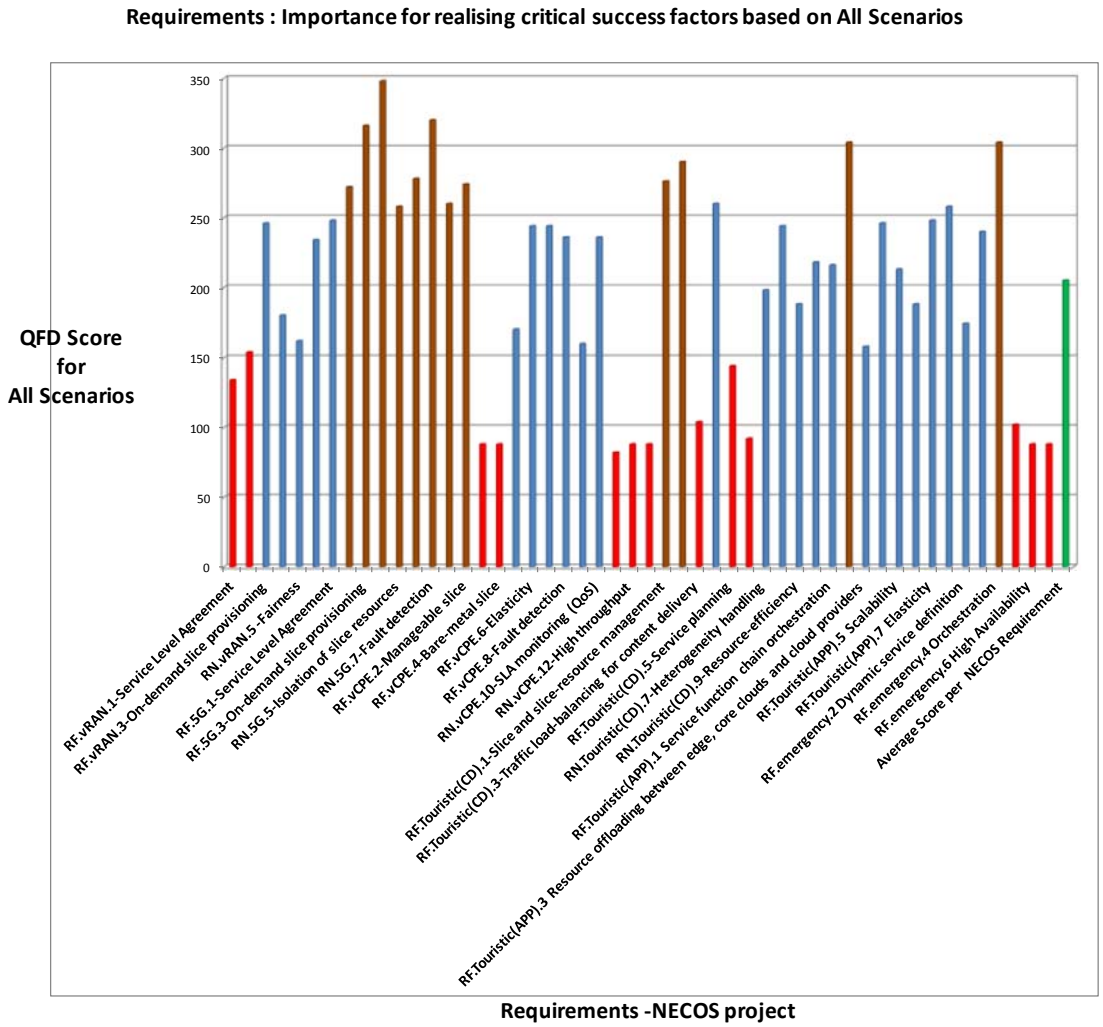

Figure A3.14. All Scenarios View Point– Realising Critical Success Factors

- 12 (of 50) requirements marked in brown colour are emerging as very important in realising Critical Success Factors as far as All Scenarios is concerned.
- 13 (of 50) requirements marked in red colour are emerging as having a lower than average importance in realising Critical Success Factors as far as All Scenarios is concerned.
- The rest of 25 (of 50) requirements marked in blue colour are emerging as having average importance in realising Critical Success Factors as far as All Scenarios is concerned.

All Scenario View Point On project Key Performance Indicators

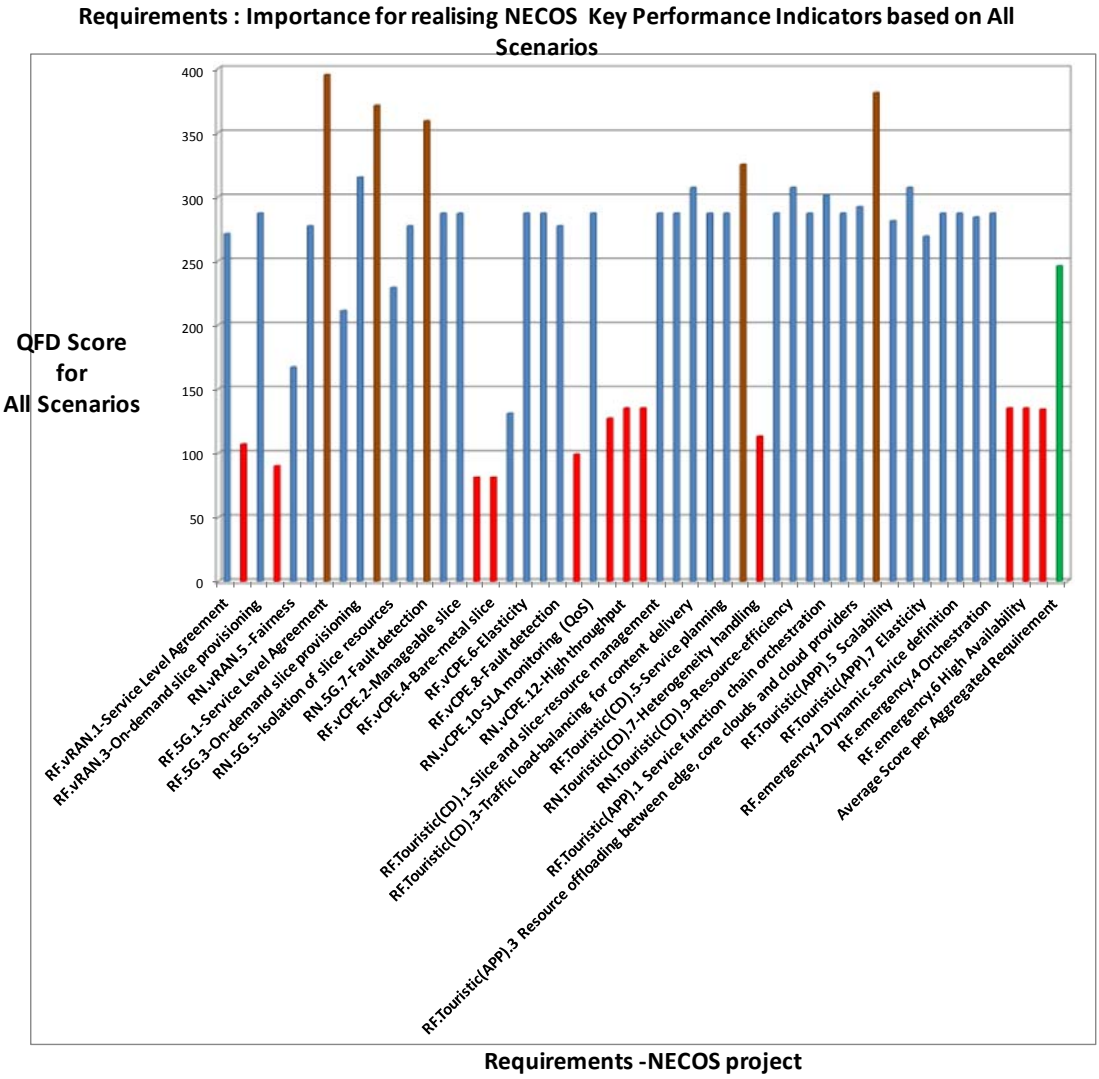

Figure A3.15. All Scenarios View Point– Realising Critical Success Factors

- 5 (of 50) requirements marked in brown colour are emerging as very important in realising Key Performance Indicators as far as All Scenarios is concerned.
- 12 (of 50) requirements marked in red colour are emerging as having a lower than average importance in realising Key Performance Indicators as far as All Scenarios is concerned.
- The rest of 32 (of 50) requirements marked in blue colour are emerging as having average importance in realising Key Performance Indicators as far as All Scenarios is concerned.

All Scenarios View Point On Project Expected Differentiated Factors (Characteristics)

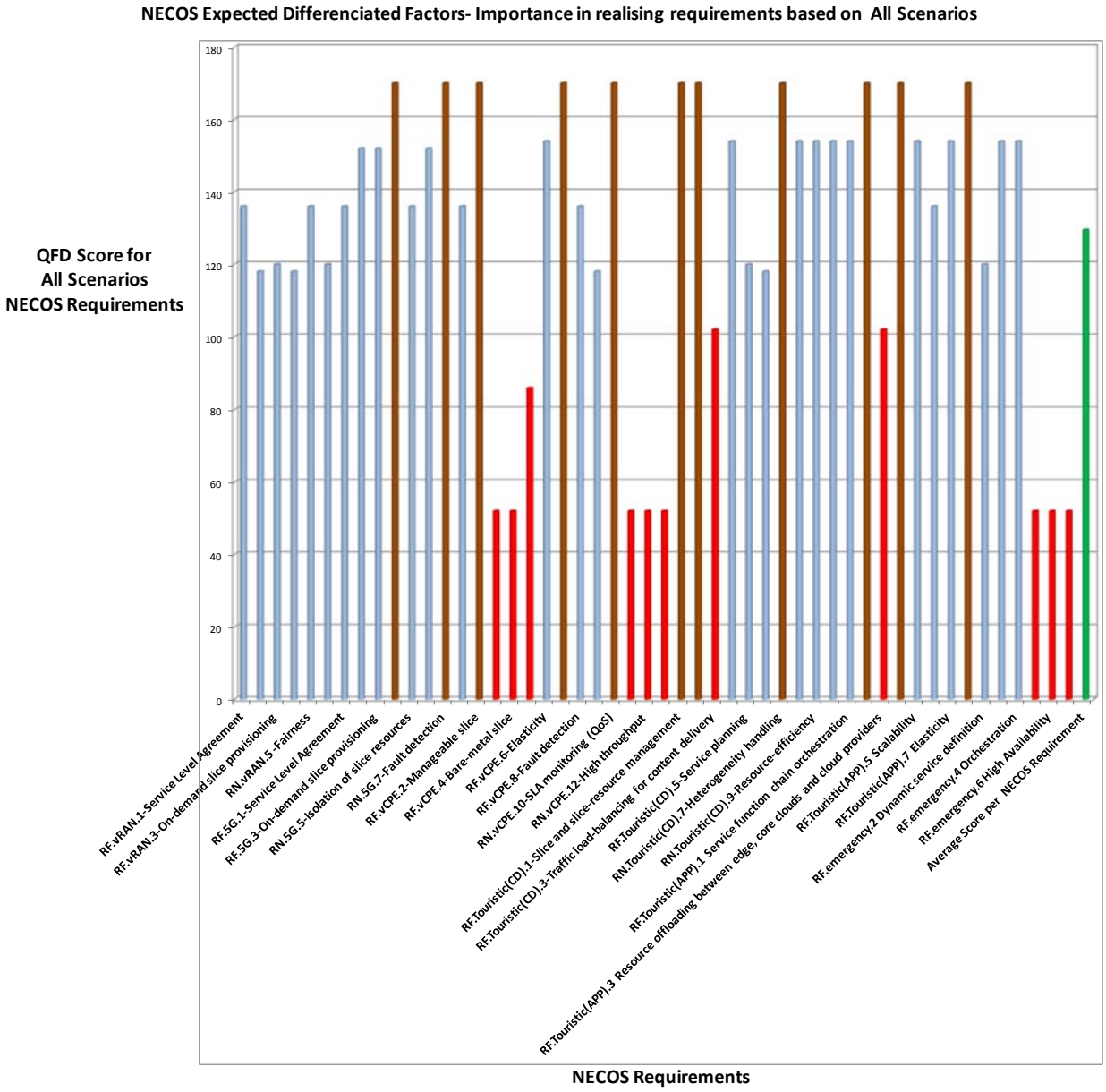

Figure A3.16. All Scenarios View Point – Realising Expected Differentiated Factors

- 11 (of 50) requirements marked in brown colour are emerging as very important in realising Expected Differentiated Factors as far as All Scenarios is concerned.
- 11 (of 50) requirements marked in red colour are emerging as having a lower than average importance in realising Expected Differentiated Factors as far as All Scenarios is concerned.
- The rest of 34 (of 50) requirements marked in blue colour are emerging as having average importance in realising Expected Differentiated Factors as far as All Scenarios is concerned.