AGH UNIVERSITY OF KRAKOW

FIELD OF SCIENCE: **ENGINEERING AND TECHNOLOGY**

SCIENTIFIC DISCIPLINE: **INFORMATION AND COMMUNICATION TECHNOLOGY**

DOCTORAL THESIS

Organization and automation of a standards-aligned, declarative Network Slice/Subnet design process

Author: **Przemysław Wyszkowski, M.Sc. Eng.**

First supervisor: **Prof. Krzysztof Zieliński, Ph.D. Eng., D.Sc.**
Assisting supervisor: **Łukasz Czekierda, Ph.D. Eng., Assoc. Prof.**

Completed in: **AGH University of Krakow,**
Faculty of Computer Science

Krakow, 2025

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

DZIEDZINA: **NAUKI INŻYNIERYJNO-TECHNICZNE**

DYSCYPLINA: **INFORMATYKA TECHNICZNA I TELEKOMUNIKACJA**

ROZPRAWA DOKTORSKA

Organizacja i automatyzacja zgodnego ze standardami, deklaratywnego procesu projektowania Network Slice/Subnet

Autor: **mgr inż. Przemysław Wyszkowski**

Promotor rozprawy: **prof. dr hab. inż. Krzysztof Zieliński**
Promotor pomocniczy: **dr inż. Łukasz Czekierda, prof. AGH**

Praca wykonana w: **Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie, Wydział Informatyki**

Kraków, 2025

"Behind a successful man, there stands a strong woman"
- a common proverb (very true in my case)

To my beloved wife Edyta and my dearest daughter Jagoda…

Acknowledgments

Without a doubt, all the scientific and organizational efforts that led to the accomplishment of this thesis have been the greatest challenge in my professional career so far. The realization of this project has taken up a large part of my life, costing me and my closest ones many sacrifices over six years of hard, painstaking and arduous work. While it brought moments of soothing satisfaction, this effort was also often paid for with tears, sleepless nights, and deteriorating mental health. Fortunately, on this long journey full of ups and downs, I was lucky to be surrounded by wonderful companions who shared many of these burdens with me. Here I have the honorable duty to pay tribute to the people who, with their commitment and support, contributed greatly to the conduct and completion of this research.

First, I would like to express my deepest gratitude to my supervisors, Prof. Krzysztof Zieliński and PhD Łukasz Czekierda. The unwavering support you have offered me in various situations throughout the long years of my PhD endeavors has been invaluable - thank you for sharing your rich experience and vast knowledge with me, as well as for keeping my research on the right track. I am especially grateful for your patience, constant readiness to help, and keeping my spirits up, which proved crucial in bringing this work to fruition.

I would like to thank all my colleagues I had the chance to cooperate with at the Samsung R&D Institute Poland and at the AGH University. Big thanks go to Jan Kienig, with whom I was fortunate to work on many interesting Network Slicing solutions and patentable ideas. I particularly enjoyed our inspiring brainstorming sessions, without which my work on this thesis would not have even commenced. Jasiek, I admire (not only me) your holistic and detailed, yet concise approach (which I clearly lack) to designing effective solutions to complex problems. You make complicated things look simple, which makes you a brilliant architect and a great teammate to cooperate with. Conducting my research would not have been possible without the engagement and intercession of my managers: Łukasz Filimowski, Marcin Jarząb, Bartłomiej Gołuszka, Szymon Stefański and Gracjan Kwiatkowski – I appreciate your understanding, patience and various forms of support you managed to provide me in the often-harsh corporate conditions. I am especially grateful for the help of Szymon Stefański, who eagerly reviewed my longish texts. I am also obliged to Piotr Chołda for insightful advices and to Mateusz Zawadzki who, at a moment when all my hope was lost and I was close to giving up, extended a much-needed helping hand that allowed us to finish our article and ultimately lead it to publication.

I am thankful to my family for the spiritual support of my efforts. Mom Krystyna, dad Stanisław, brother Grzegorz, as well as my parents-in-law Krystyna and Lesław – thank you for the understanding you have shown me and all the support you offered me throughout the many years of my endavours. I am grateful to my friends, especially Maciej Matoszko and Mateusz Hycz, who strived on lifting my spirits, encouraging me to persevere in pursuing my goals – you are always there when I need you. Apart of thanking, I also need to sincerely apologize both my family and friends, for whom I was frequently unavailable over the last years. I apologize for all the important events I missed, and all the lost time that we should have spent together – I hope that the “after-PhD” era will allow us to make up for it.

The greatest thanks, and at the same time, the greatest apologies for all the time I took from them, are due to my dearest daughter Jagoda and my beloved wife Edyta. I owe you a huge debt of attention and dedication that I am obliged to repay. Truly, the fulfillment of my ambitions costed you a lot of sacrifices, requiring loads of patience and understanding. Without your encouragement and limitless support, it would not be possible to conduct and complete my research. Thank you for your unconditional love, trust and faith in me. You are my true angels and I love you with all my heart. Finally, I would like to thank God for the gift of life and all the blessings He has bestowed upon me – for favoring me and allowing me to develop, for surrounding me with good-hearted people and for keeping me and my closest ones in good health.

Przemysław Wyszkowski,
Warsaw, May 2025

ABSTRACT

According to the original visions, the technology of Network Slicing should allow for leveraging the fifth-generation telecommunication networks (5G) infrastructure for dynamic creation of multiple virtual, lightweight, but at the same time fully fledged telecommunication networks called Network Slice Instances (NSI) and their logical parts called Network Slice Subnet Instances (NSSI). Importantly, NSIs/NSSIs are expected to provide communication services with guaranteed Quality of Service (QoS) tailored to the Service-Level Agreement (SLA) requirements of business services running on top them, as well as the devices and end users accessing them. Although set up on a virtualized and shared cloud infrastructure, all NSIs and NSSIs should be isolated from each other, both in terms of the characteristics of transferred data and performance, as well as security and management. It is also expected that both NSIs, NSSIs, and the elements from which they are built, should be shared in various combinations to optimize the use of the telco infrastructure, and that they should be flexibly created on demand and dynamically adapted to the Customers' requirements.

Although intensive research and standardization work has been carried out since the formulation of the basic concepts of Network Slicing in 2015, the adoption of this technology by the telco Providers is progressing slowly and, so far, in limited scope in relation to the originally assumed properties. This concerns, among others, the dynamicity and flexibility of creating NSI/NSSI, as well as the possibility of their utilization by a wide range of target Customers, including small and medium-sized enterprises or individual Customers. Among the factors contributing to this state is the highly fragmented ecosystem of bodies standardizing this technology and the lack of their coordinated work, resulting in incomplete coverage of many important technological aspects. One of such critical areas is the design process of NSI/NSSI and, on this basis, the allocation of an appropriate share of its resources to ensure that the QoS/SLA requirements of the Customer ordering its creation are satisfied.

The currently leveraged methods of designing and allocating NSIs/NSSIs are based on so-called *imperative provisioning style*. Such approach assumes direct involvement of a human Operator in the design process, who, based on the gained domain and operational experience, decides on reusing an already existing NSI/NSSI or creating a new one. During the design process, the Operator needs to determine the NSI/NSSI structure and the specific configuration of the elements it consists of, the amount of resources consumed by them, and make decision about allocating a specific share of their capabilities that will allow for meeting the Customer's QoS/SLA requirements. The need of human Operator's involvement in complex decision-making that takes place during the design process carries a number of negative consequences, such as: limited possibilities of delivering NSI/NSSI on demand, inflated operational expenditures (OPEX), susceptibility to errors, as well as limited accessibility of this technology to Customers, which are narrowed to large enterprises that can afford direct cooperation with telco Providers at the business and technical levels.

In order to limit the Operator's involvement in the abovementioned complex decision-making processes, the current telco management and orchestration standards assume the so-called *declarative provisioning style* of NSIs/NSSIs. According to such an approach, NSI/NSSI allocation is possible to be triggered programmatically directly by the Customers, who only need to provide high-level SLA/QoS requirements to be met by the NSI/NSSI to be delivered, without specifying any technical details regarding its structure or configuration of its internal elements – it is the Network Slicing management and orchestration system's responsibility to devise the specification of a suitable NSI/NSSI that would meet the posed requirements. However, apart from the definition of the declarative style operations' signatures, the standards do not provide clearly defined models or procedures enabling their effective implementation. In particular, there are currently no standardized design-time models for NSI/NSSI or its internals. There are also no standardized procedures that would allow for the transformation of the Customer's high-level QoS/SLA requirements into the technical specification of an NSI/NSSI that provides the possibility of satisfying them.

The main motivation of the work described in this dissertation is to fill the abovementioned gaps by proposing solutions improving the NSI/NSSI design process, so that the entire, end-to-end procedure of its allocation is fully automated, both from the perspective of the Customer, the Operator and the Provider. The author believes that by providing an easy-to-use, declarative way of NSI/NSSI allocation that is supported by fully automated mechanisms of their design, the Network Slicing technology has a chance to be offered with the originally assumed properties, where NSIs/NSSIs are delivered on demand, in massive quantities, and utilized by a wide range of Customers. Thus, the problem of organizing and automating a Network Slice/Subnet design process undertaken in this dissertation is an important issue not only from a cognitive, but also from a practical and business perspective.

The author puts forward a thesis that modeling the Operator's and Provider's knowledge, uniform modeling of the NSI/NSSI building blocks and the structuralization of the design activities to leverage the modeled knowledge allows for defining a coherent and generic conceptual framework of a design process, which enables construction of multivariant and extendable systems performing automated transition from high-level Customer's requirements to a technical specification of an NSI/NSSI capable of meeting them. In order to validate the formulated thesis, the problem of elaborating design-time models and methods for automating the complex NSI/NSSI design process that meets the assumptions of the *declarative provisioning style* is undertaken, with full alignment to the management and orchestration standards in force. To achieve the assumed goals, the author begins by conducting a broad gap analysis of existing telco networks management standards and the NSI/NSSI provisioning methods proposed in research works, making an orderly systematization of models, systems and approaches to design-time issues. Based on the identified gaps, the author formulates requirements to be met by the sought solution, constituting a determinant for the elaborated concepts.

Next, in the conceptual part of the dissertation, the author proposes an original, holistic taxonomy of elements out of which an NSI/NSSI may be designed, which are referred to as *Slicing Entities*. As a part of the elaborated taxonomy, several abstractions are proposed that allow for uniform modeling of such elements from the perspective of characteristics important to be considered during the design process (*Slicing Entity Capabilities*) and for the encapsulation of coherent sets of values for configuring such elements (*Slicing Entity Configuration Profiles*). Then, models and methods enabling automated processing of requirements of different abstraction levels are proposed, including their decomposition and accumulation, as well as aggregation of *Slicing Entities*' *Capabilities* and their consumption by the posed requirements. Furthermore, modeling constructs conveying the design policies agreed between the Customer and the Provider are provided, as well as a fully standards-aligned NSI/NSSI design-time model. Finally, the overall procedural framework of the design process is specified, consisting of high-granularity activities with clearly defined input/output and complementary roles, along with the definition of gradual, backward-compatible roadmap for implementing NSI/NSSI design automation in the target management and orchestration systems.

The proposed conceptual framework is validated in the practical part of the dissertation. A broad description of a Proof of Concept (PoC) system, which implements a selected scope of the proposed framework, presents its feasibility, demonstrating a number of its properties, such as coherence, multivariant nature and extendability. The evaluation of the PoC solution from a functional perspective confirms that it generates the expected results in a representative set of operational conditions, while not requiring the Operator to be involved in the design and allocation process. The evaluation from a performance perspective shows fully satisfactory times of producing fully concretized design-time descriptions based solely on the Customer's high-level SLA/QoS requirements. The obtained results confirm the possibility for leveraging the presented solutions for dynamic delivery of NSIs/NSSIs on demand, in large quantities and at various scales. The entirety of the collected experimental results allows for the confirmation of the validity of the formulated thesis, exhibiting high potential for practical implementation of the elaborated models and methods, as well as their standardization.

STRESZCZENIE

Według oryginalnych wizji, technologia Network Slicingu ma pozwalać na wykorzystanie infrastruktury telekomunikacyjnej piątej generacji (5G) do dynamicznego tworzenia wielu wirtualnych, lekkich, ale jednocześnie w pełni funkcjonalnych sieci telekomunikacyjnych zwanych Network Slice Instance (NSI) oraz ich logicznych części zwanych Network Slice Subnet Instance (NSSI). Co istotne, NSI/NSSI mają zapewniać usługę telekomunikacyjną z gwarancjami jakości (ang. Quality of Service, QoS) indywidualnie dostosowanymi do wymagań Service-Level Agreement (SLA) aplikacji biznesowych w nich działających oraz urządzeń i użytkowników z nich korzystających. Mimo że zestawiane na zwirtualizowanej i współdzielonej infrastrukturze chmurowej, wszystkie NSI i NSSI powinny być od siebie odizolowane, zarówno pod względem charakterystyki i wydajności przesyłanych danych, jak i bezpieczeństwa oraz zarządzania. Ponadto oczekuje się, że zarówno NSI, NSSI, jak i elementy, z których są one budowane, powinny być współdzielone w różnych kombinacjach dla zoptymalizowania wykorzystania infrastruktury telekomunikacyjnej, oraz że będą elastycznie tworzone na żądanie i dynamicznie dostosowywane do wymagań klientów.

Mimo, że od momentu sformułowania podstawowych koncepcji Network Slicingu w roku 2015 prowadzone są intensywne prace badawcze i standaryzacyjne, praktyczne wdrożenie tej technologii przez dostawców telekomunikacyjnych postępuje wolno i - jak dotychczas - w ograniczonym zakresie w stosunku do oryginalnie zakładanych własności. Dotyczy to m.in. dynamizmu i elastyczności tworzenia NSI/NSSI, jak również możliwości ich wykorzystania przez szeroki wachlarz klientów docelowych, łącznie z małymi i średnimi przedsiębiorstwami czy klientami indywidualnymi. Pośród czynników wpływających na taki stan jest silnie sfragmentowany ekosystem organizacji standaryzujących tę technologię i brak ich skoordynowanego działania, skutkujący wciąż niepełnym pokryciem wielu ważnych obszarów technologicznych. Jednym z takich obszarów jest projektowanie NSI/NSSI i, na tej podstawie, alokacja odpowiedniej części zasobów elementów z których się składają dla zapewnienia spełnienia wymagań QoS/SLA klientów zlecających ich utworzenie.

Obecnie stosowane metody projektowania i alokacji NSI/NSSI są oparte o tzw. „imperatywny styl dostarczania" (ang. *imperative provisioning style*). Takie podejście zakłada zaangażowanie w proces projektowania operatora sieci telekomunikacyjnej – eksperta, który w oparciu o swoje doświadczenie, wiedzę domenową i operacyjną, decyduje o wykorzystaniu już istniejącego NSI/NSSI albo o stworzeniu nowego. Podczas projektowania, operator wyznacza topologię i strukturę NSI/NSSI, konfiguracje elementów z których się składa, ilość konsumowanych przez nie zasobów, oraz decyduje o zaalokowaniu określonej części ich możliwości dla zaspokojenia wymagań QoS/SLA klienta. Konieczność zaangażowania człowieka w skomplikowane procesy decyzyjne występujące podczas projektowania niesie szereg negatywnych konsekwencji, takich jak: ograniczone możliwości dostarczania NSI/NSSI na żądanie, zawyżone koszty operacyjne (OPEX), podatność na błędy, jak również zawężenie grona klientów jedynie do dużych podmiotów, które stać na bezpośrednią współpracę z dostawcami na poziomie biznesowym i technicznym.

By ograniczyć udział operatora w wyżej wymienionych, złożonych procesach decyzyjnych, obecne standardy telekomunikacyjne zakładają tzw. „deklaratywny styl dostarczania" NSI/NSSI (ang. *declarative provisioning style*). Według takiego podejścia, alokacja NSI/NSSI może być zlecona programistycznie bezpośrednio przez klienta, który dostarcza jedynie wysokopoziomowe wymagania SLA/QoS do spełnienia, bez określania jakichkolwiek szczegółów technicznych dotyczących budowy NSI/NSSI czy konfiguracji jego elementów składowych – obowiązek sporządzenia projektu odpowiedniego NSI/NSSI jest przeniesiony na zautomatyzowany system zarządzania. Jednakże, poza definicją sygnatur deklaratywnych operacji, standardy nie specyfikują modeli ani procedur umożliwiających ich efektywną implementację. W szczególności, obecnie nie istnieją żadne standardy modeli NSI/NSSI i ich elementów składowych w czasie projektowania (ang. *design-time models*). Nie są również sprecyzowane procedury, które pozwalałyby na transformację wysokopoziomowych wymagań QoS/SLA Klienta do specyfikacji technicznej NSI/NSSI dającego możliwość ich zaspokojenia.

Główną motywacją prac opisanych w niniejszej rozprawie jest wypełnienie wymienionych braków poprzez zaproponowanie rozwiązań, które usprawnią procesy projektowania NSI/NSSI tak, by całościowy proces ich alokacji był w pełni zautomatyzowany, zarówno z perspektywy klienta, jak i operatora i dostawcy. Autor uważa, że poprzez udostępnienie łatwego w wykorzystaniu interfejsu tworzenia NSI/NSSI wspartego przez w pełni zautomatyzowane mechanizmy ich projektowania, technologia Network Slicingu ma szansę być oferowana zgodnie z oryginalnie zakładanymi właściwościami, gdzie NSI/NSSI są dostarczane na żądanie, w znacznych ilościach i możliwe do wykorzystania przez szeroki wachlarz klientów. Stąd podejmowana w tej dysertacji tematyka zorganizowania zautomatyzowanego procesu projektowania NSI/NSSI jest ważnym zagadnieniem nie tylko z perspektywy poznawczej, ale również praktycznej i biznesowej.

Autor stawia tezę, że modelowanie wiedzy domenowej i operacyjnej operatora, jednorodne modelowanie elementów składowych NSI/NSSI oraz odpowiednia strukturalizacja aktywności projektowania dla efektywnego wykorzystania zamodelowanej wiedzy pozwala na zdefiniowanie ogólnych i spójnych ram konceptualnych procesu projektowania, według których możliwe jest konstruowanie wielowariantowych i rozszerzalnych systemów automatyzujących transformację wysokopoziomowych wymagań klienta do specyfikacji technicznej NSI/NSSI zdolnego do ich spełnienia. By wykazać sformułowaną tezę, autor podejmuje zadanie opracowania modeli czasu projektowania i metod automatyzacji złożonego procesu projektowania NSI/NSSI spełniającego założenia deklaratywnego dostarczania, z wymogiem pełnej zgodności z obowiązującymi standardami zarządzania sieciami telekomunikacyjnymi. Dla osiągnięcia zakładanego celu, autor rozpoczyna od przeprowadzenia szerokiej analizy luk (ang. *gap analysis*) w istniejących standardach zarządzania sieciami telekomunikacyjnymi i obecnie stosowanych metodach dostarczania NSI/NSSI proponowanych w pracach badawczych, dokonując uporządkowanego zestawienia modeli, systemów oraz podejść do zagadnień czasu projektowania w obszarze Network Slicingu. W oparciu o zidentyfikowane luki, autor formułuje wymagania wobec poszukiwanego rozwiązania, stanowiące wyznacznik dla opracowywanych koncepcji.

Następnie, w części koncepcyjnej pracy, proponowana jest autorska, holistyczna taksonomia elementów z których można projektować NSI/NSSI, określanych przez autora mianem *Slicing Entities*. W ramach opracowanej taksonomii proponowane są abstrakcje pozwalające na jednolite modelowanie takich elementów, m.in. z perspektywy właściwości istotnych do uwzględnienia podczas projektowania (*Slicing Entity Capabilities*) oraz spójnych zestawów konfigurujących takie elementy (*Slicing Entity Configuration Profiles*). Ponadto, definiowane są modele i metody umożliwiające zautomatyzowane przetwarzanie wymagań o różnych poziomach abstrakcji, ich dekompozycji i akumulacji, jak również agregacji własności *Slicing Entities* i ich konsumpcji przez stawiane wymagania. Proponowane są również konstrukcje modelujące uzgodnione między klientem i dostawcą polityki projektowania oraz definiowana jest struktura w pełni zgodnego ze standardami modelu opisującego NSI/NSSI w czasie projektowania. Finalnie, precyzowane są całościowe ramy proceduralne procesu projektowania, złożonego z wysokogranularnych aktywności o jasno określonym wejściu/wyjściu i komplementarnych wobec siebie rolach, wraz z określeniem stopniowych, kompatybilnych wstecznie możliwości wdrożenia automatyzacji projektowania NSI/NSSI w systemach zarządzania i orkiestracji sieci 5G.

Opracowane koncepcje modeli i metod projektowania zostają poddane weryfikacji w części praktycznej dysertacji. Szeroki opis implementacji systemu opartego o proponowane ramy konceptualne, prezentuje ich realizowalność, wykazując jednocześnie szereg ich własności, takich jak spójność, wielowariantowość i rozszerzalność. Badania eksperymentalne zostały przeprowadzone na danych konfiguracyjnych zbliżonych do rzeczywistych, wykazując poprawność działania systemu w szeregu uwarunkowań operacyjnych, przy jednoczesnym braku konieczności zaangażowania operatora w proces projektowania i alokacji. Ewaluacja z perspektywy wydajnościowej wykazuje w pełni satysfakcjonujące czasy projektowania na bazie wejściowych wymagań klienta, pozwalając na zastosowanie przedstawionych rozwiązań dla dynamicznego dostarczania NSI/NSSI na żądanie, w dużych ilościach i w różnych skalach. Całość zgromadzonych rezultatów eksperymentalnych pozwala na potwierdzenie zasadności sformułowanej tezy, wykazując jednocześnie wysoki potencjał dla praktycznego wdrożenia opracowanych rozwiązań i ich standaryzacji.

Table of Contents

Acknowledgments ***3***
Abstract ***5***
Streszczenie ***7***
Table of Contents ***9***
Chapter 1. *Introduction* ***13***
1.1. Research area introduction – Network Slicing **14**
1.2. Motivation **15**
1.3. Problem statement **18**
1.4. Thesis statements and dissertation goals **20**
1.5. Contributions **20**
1.6. Scope **22**
1.7. Structure **25**
Chapter 2. *Background* ***27***
2.1. 5G basics **27**
2.1.1. Requirements and expectations 27
2.1.2. Key architectural principles of 5G 28
2.1.3. 5G functional architecture 29
2.1.4. Organization of 5G Network Functions and Links in the RAN domain 30
2.1.5. Organization of 5G Network Functions and Links in the Core domain 31
2.1.6. Network Slicing as a feature of 5G 32
2.2. Basic aspects of standards-aligned management of 5G networks **33**
2.2.1. Conceptual models of elements comprising a 5G network 33
2.2.2. Management models of elements comprising a 5G network 34
2.2.3. Design-time models of elements comprising a 5G network 37
2.2.4. Systems utilized for 5G networks management and orchestration 40
2.2.5. Basic management procedures related to individual or groups of elements comprising a 5G network 43
2.3. Basic aspects of standards-aligned Network Slicing management **45**
2.3.1. Basic Network Slicing management concepts 46
2.3.2. Input triggering NSI/NSSI design and allocation - SLA vs SLS 48
2.3.3. Network Slicing management models 50
2.3.4. Lack of standardized Network Slicing design-time models 51
2.3.5. Standardized Network Slicing management stack and gaps within it 54
2.3.6. NSI/NSSI offering modes - relation to static and dynamic slicing, provisioning styles and design automation 56
2.4. Requirements to be met by a generic Network Slice/Subnet design automation solution **57**
2.5. Related work **58**
2.5.1. Overview of related research 58
2.5.2. Imperative style approaches 59
2.5.3. Declarative style approaches 61
2.5.4. Approaches attempting standards alignment 63
2.6. Summary **65**
Chapter 3. *Network Slicing design-time concepts, definitions and modeling constructs* ***67***
3.1. Modeling of Slice/Subnet building blocks – Slicing Entities **67**
3.1.1. Slicing Entities taxonomy 68
3.1.2. Slicing Entity Capabilities 72
3.1.3. Slicing Entity Configuration Profiles 74
3.1.4. Interpretation of Slicing Entity Capabilities' values 75

3.2. Definitions of fundamental Network Slicing design-time aspects ... 76
3.2.1. Definition of the Network Slice/Subnet design process ... 77
3.2.2. Definition of the Network Slice/Subnet design outcomes ... 77
3.3. Modeling of Slice and Subnet *Instance Descriptors* ... 78
3.3.1. Role of a Slice/Subnet Instance Descriptor ... 78
3.3.2. Determination of information items pertaining to the internals of the designed NSI/NSSI ... 79
3.3.3. Determination of information items pertaining to the designed NSI/NSSI as a whole ... 83
3.3.4. Summary of information items to be contained in a Slice/Subnet Instance Descriptor ... 84
3.3.5. Layered distribution of information items among 3GPP and NFV-MANO Descriptors ... 84
3.3.6. Proposed structure of 3GPP-aligned Slice/Subnet *Instance Descriptors* ... 85
3.4. Modeling of individual SLS attributes and their handling ... 86
3.5. Modeling of SLS handling in the Slice/Subnet internal structure ... 87
3.5.1. Modeling of *Requirements* decomposition ... 87
3.5.2. Modeling of *Capabilities* aggregation ... 89
3.6. Modeling of operational and business objectives ... 90
3.7. Summary ... 90
***Chapter 4.** Conceptual framework of a Network Slice/Subnet design process and its automation ... 93*
4.1. Preliminary remarks ... 93
4.2. High-level overview ... 94
4.2.1. Inputs to the design process ... 95
4.2.2. Walkthrough of the inherent design activities ... 95
4.2.3. Walkthrough of the post-design activities ... 96
4.3. Logic and automation of activities determining NSI/NSSI internal structure ... 97
4.3.1. Allocation request analysis ... 97
4.3.2. Initial evaluation of candidates (representing Slice/Subnet as a whole) ... 97
4.3.3. Selection of candidates ... 99
4.3.4. Requirements decomposition ... 100
4.3.5. Subsequent evaluation and selection of candidates (low-level Slicing Entities, deployment environments) ... 101
4.3.6. Configuration of internals ... 102
4.4. Logic and automation of activities determining Slice/Subnet-level characteristics ... 103
4.4.1. Determining Slice/Subnet-level Configuration Profiles and aggregated Capabilities ... 103
4.4.2. Determining Slice/Subnet-level SLS-assurance configuration ... 104
4.5. Logic and automation of activities finalizing the design outcomes ... 105
4.5.1. Validations ... 105
4.5.2. Instance Descriptors serialization and packaging ... 105
4.5.3. Determining the share of Capabilities to be allocated ... 106
4.6. Logic and automation of post-design activities ... 106
4.6.1. Pre-provisioning of design outcomes ... 107
4.6.2. Closed-loop feedback ... 107
4.6.3. Triggering NSI/NSSI provisioning and allocation procedures ... 108
4.7. End-to-end design process automation ... 108
4.7.1. Assumptions simplifying achieving end-to-end automation ... 108
4.7.2. Roadmap for gradual rollout of design process automation ... 109
4.7.3. Desired level of automation ... 111
4.7.4. Design process duration and dynamicity ... 112
4.8. Summary ... 113
***Chapter 5.** Proof of Concept implementation ... 115*
5.1. PoC assumptions and expectations ... 115
5.1.1. PoC goals and research methodology ... 115
5.1.2. PoC implementation requirements and scope ... 116
5.1.3. PoC caveats ... 117
5.2. Choice of a Subject Orchestrator ... 118
5.2.1. Preliminary remarks and survey assumptions ... 118
5.2.2. Considered open-source Orchestrators ... 118

5.2.3. Possibility of utilizing a proprietary Orchestrator prototype 119
5.3. Overview of the Subject Orchestrator, before PoC implementation 120
5.3.1. Subject Orchestrator's original architecture 120
5.3.2. Subject Orchestrator's operational scope and limitations 122
5.3.3. Characteristics of MFs supported by the Subject Orchestrator 123
5.4. Shortcomings of Subject Orchestrator targeted for improvement in the PoC 125
5.4.1. Shortcoming 1 - Manual NSSI design and triggering of its provisioning by the Operator 125
5.4.2. Shortcoming 2 - Allocation of existing NSSIs only, selected by Customer 127
5.4.3. Shortcoming 3 - Limited set of supported high-level SLS attributes 128
5.4.4. Summary of shortcomings targeted in the PoC 129
5.5. Implementation of the individual framework's elements 130
5.5.1. ASD PoC elements in the Subject Orchestrator's architecture 130
5.5.2. Models' implementation 131
5.5.2.1. Models of abstract Slicing Entities and their concretizations 132
5.5.2.2. Models of Deployment-time and Runtime Slicing Entities 133
5.5.2.3. Models for management of SLS Requirements and Capabilities 134
5.5.2.4. Organization of SLS management objects in a sample scenario 136
5.5.2.5. Models of Design-time Slicing Entities 137
5.5.2.6. Models conveying results of the design activities 139
5.5.2.7. Design process policies model 140
5.5.3. Individual design activities' implementation 141
5.5.3.1. Evaluation 141
5.5.3.2. Selection 142
5.5.3.3. Requirements decomposition 142
5.5.3.4. Capabilities aggregation 142
5.5.3.5. InstanceDescriptor generation 142
5.5.3.6. Determination of Capabilities share for supporting the input SliceProfile 143
5.5.3.7. Optimization of processing in selected design activities 143
5.6. Implementation of the ASD end-to-end NSSI design, provisioning and allocation workflow 143
5.6.1. Organization of the ASD workflow 143
5.6.2. Prerequisites for automated execution of the ASD workflow 144
5.6.3. Automated SliceProfile validation and decomposition of high-level SLS Requirements 145
5.6.4. Automated evaluation and selection of an existing NSSI 148
5.6.5. Automated design of a new NSSI 149
5.7. PoC implementation summary 150
5.7.1. Achieved functional augmentation 150
5.7.2. Conclusions related to the technical realization of the conceptual framework 151
5.7.3. PoC implementation enhancements considered as a future work 152
***Chapter 6.** Experimental evaluation 155*
6.1. Goals of PoC experimental evaluation 155
6.2. Organization of the experiments 155
6.2.1. Scope 155
6.2.2. Methodology 156
6.2.3. Assumptions and caveats 157
6.2.4. Testbed 157
6.3. Part 1 - functional evaluation – a Case Study 159
6.3.1. Operational scenery 159
6.3.2. Operational conditions of the experiments 160
6.3.2.1. Control-plane of the RAN infrastructure – gNB MEs synthetic data 160
6.3.2.2. DesignPolicies 162
6.3.2.3. NSSTs 163
6.3.2.4. State of operational objects 163
6.3.3. Scenarios and Test Cases specification 164
6.3.4. Selected details of Case Study execution and results analysis 166
6.3.4.1. "Rainy-day" TCs in Scenario 1 166
6.3.4.2. "Sunny-day" TC in Scenario 1 167

6.3.4.3. Scenario 2 ... 171
6.3.4.4. Scenario 3 ... 174
6.3.4.5. Scenario 4 ... 176
6.3.4.6. Scenario 5 ... 178
6.3.5. Functional evaluation conclusions ... 180
6.4. Part 2 - performance evaluation ... 182
6.4.1. Clarification of experiments' scope and assumptions ... 182
6.4.2. Conditions of the experiments ... 183
6.4.2.1. Control-plane of the RAN infrastructure – gNB NEs synthetic data ... 183
6.4.2.2. Subject Orchestrator's load conditions - scale of NSSIs already allocated ... 185
6.4.3. Test Cases specification ... 186
6.4.4. Selected details of Test Cases execution ... 187
6.4.5. Obtained results and their analysis ... 188
6.4.5.1. Allocation of new, not-loaded NSSIs ... 189
6.4.5.2. Allocation of new, loaded NSSIs ... 192
6.4.5.3. Allocation of existing NSSIs ... 193
6.4.6. Performance evaluation conclusions ... 195
***Chapter 7.** Conclusions ... 197*
7.1. Thesis verification ... 197
7.2. Discussion on the industrial applicability of the proposed concepts and their potential impact ... 198
7.3. Future work ... 200
7.4. Anticipated research directions in the Network Slicing design-time area ... 201
Appendix: Organization of the 3GPP and ETSI NFV-MANO standards ... 203
List of Acronyms ... 204
List of Figures ... 205
List of Charts ... 206
List of Sequence Diagrams ... 206
List of Tables ... 206
***References** ... 207*

Chapter 1

Introduction

Since the inception of the first-generation telecommunication networks (1G) in the mid-70s of the 20th century, a continuous evolution of the technologies underpinning them has been observed. On the one hand, it is driven by the increasingly ubiquitous availability of the **Communication Services** (CS) offered by these networks (i.e. voice, Internet access, text, video, etc.), the expansion and proliferation of business **Services** relying on them, the ever-increasing demands of the Communication Service **Customers** (CSC, Customer) utilizing the CSs, as well as its increasingly strategic role. On the other hand, this evolution is also sparked through the application of technological advancements that open up new possibilities of telecommunication networks exploitation and their more efficient operation by the Communication Service **Providers** (CSP, Provider). To keep up with the growing expectations and to provide the means for new use cases to emerge, the telecommunication technology is a subject of intensive research, development and standardization. The currently observed, major technological shift is related to the transition from the 4th Generation (4G) to the 5th Generation (5G) telecommunication networks, as well as the already ongoing research towards their 6th Generation (6G).

At present, the majority of Providers around the globe exploit the 4G-Advanced and initial deployments of the 5G networks operated in the Non-Standalone[1] mode (5G-NSA). Typically, Providers operate such infrastructure according to the "one size fits all" philosophy, offering a range of CSs by means of a single, common network operated on the "best effort" basis. On top of these CSs, a large variety of Services are offered and accessed by various Customers who connect to the Provider's network via User Equipment (UE)[2] devices. However, while many of the currently offered Services run with satisfactory quality and performance, the "best effort" operation of such networks is unable to satisfy the diverse requirements of the emerging use cases originally envisioned to be supported by the 5G and the future 6G networks. Such use cases include Vehicle-to-Everything (V2X) solutions, Augmented Reality (AR), SmartCity or SmartFactory, massive Internet of Things (mIoT), Private 5G and others. For their proper functioning, Services of such types require CSs **capable of guaranteeing Quality of Service (QoS) measures**, e.g. latency not exceeding a given threshold or bandwidth with a guaranteed minimal value. Moreover, special operational characteristics are required by such Services, e.g. support for high density of UEs, high speed and 3D mobility (e.g. trains, cars, Unmanned Aerial Vehicles (UAV)) or stringent availability and reliability requirements. Offering such Services based on CSs within the today's "one size fits all" networks would not only prevent them from delivering the expected functionalities with the desired quality, but could even pose a threat to the health and life of the users, e.g. while coordinating the city traffic, conducting remote surgeries, etc.

Furthermore, in many use cases, the wide-area and permanent availability offered by CSs of a common network is not essential, because a given Service should only be provided in selected areas and/or only for a limited duration (e.g. a week, day or even for few minutes). Such narrowed areas may be of various scales, with uniform or disjointed coverage, e.g. nationwide coverage of highways for autonomous driving, city-wide coverage for localized SmartCity solutions, hotspot-wide coverage for a massive event on a stadium or

[1] 5G Non-StandAlone (NSA) mode refers to a mixed deployment of a telecommunication network consisting of 5G and 4G elements. In contrast, the StandAlone (SA) mode deployment consists of 5G elements only.

[2] Although the name User Equipment (UE) may suggest that it is an appliance utilized by a human user (like a smartphone, tablet, laptop, etc.), it is a notion utilized in the telco domain to refer to any device that is equipped with a radio interface and a Subscriber Identity Module (SIM). As such, the notion of a UE includes all devices that connect to the Provider's network, including semi or fully autonomous devices like sensors, cameras, smart fridges or washing machines, robots, Unmanned Aerial Vehicles (UAV), cars, etc.

sparse-spots coverage for smooth teleconferencing, gaming session or workgroup cooperation between a limited count of participants. Establishing dedicated or leased telecommunication networks to guarantee QoS for each of such diverse Services, although possible, is not only **too slow** and **inflexible**, but also **too costly** to be economically viable.

1.1. Research area introduction – Network Slicing

Network Slicing is among the core technologies expected to address the aforementioned concerns in the 5G and in the future 6G networks. According to the original visions described in [1] and [2], Network Slicing promises to enable spawning and operating multiple virtual and lightweight but, at the same time, full-blown telecommunication networks called Network Slice Instances (NSI) and their logical parts called Network Slice Subnet Instances (NSSI). The goal of creating and operating NSIs and NSSIs is to provide dedicated telecommunication networks in which CSs meeting **specific QoS and Quality of Experience** (**QoE) requirements** of the envisioned Services and the users of such Services can be offered. To be economically viable, NSIs and NSSIs are meant to be composed of fully virtualized internal elements that can be flexibly deployed in common, shared cloud Data Centers (DC) utilizing commodity-of-the-shelf (COTS) hardware infrastructure. Despite this fact, they shall be isolated from each other (in terms of performance, security and management [3] in the case of "Private 5G") and, most importantly, provide end-to-end QoS/QoE guarantees catered to the specific business requirements and use cases[3]. NSIs and NSSIs are expected to be created in various scales of geographical coverage, with various duration of operation (short-lived, accessible in periodic timeslots, permanent) and in potentially massive quantities. Moreover, both NSIs and NSSIs are expected to be flexibly created on demand [4] and be operated largely hands-free, leveraging closed-loop adaptation to the experienced conditions [5]–[7].

The envisioned features enumerated above make Network Slicing not only suitable to address the concerns mentioned earlier, but are also expected to open a range of new technological, research and business opportunities for the Providers, as well as for a wide range of Customers, including vertical industries[4], corporations, researchers and, most notably, smaller-scale enterprises and individual users. The Providers are promised to benefit from the economies of scale, allowing them to run 10s, 100s or even 1000s of NSIs and NSSIs deployed on the virtualized DC infrastructure, enabling them to maximize resources utilization and to reduce the capital expenditures (CAPEX). The Customers, on the other hand, are promised not only to flexibly create and utilize dedicated CSs adjusted to the specific QoS/QoE requirements of their Services, but may also become "owners" of customized 5G networks, offered by the Providers in the Network Slice as a Service (NSaaS) model [8]–[11]. Such model is envisioned to enable establishing so-called "Private 5G" networks, which may serve as an alternative to Virtual Private Networks (VPN) or as a replacement of industrial networks, among others [12], [13]. Network Slicing is also perceived as a cornerstone for the realization of the envisioned 6G use cases [14]–[17]. For all these reasons, many view Network Slicing as a key technology for 5G and 6G economic success [18]–[21] with the market worth expected to grow from US$309 million in 2022 to US$24 billion in 2028 [22].

However, despite an enormous interest in recent years from both industry and academia, many of the enumerated Network Slicing features are still in the realm of buzzwords rather than concretely achievable capabilities. This viewpoint is confirmed by the current status of commercial inception of this technology, which is broadly admitted to be progressing slowly, with the initial attempts being far from fully realizing the visions discussed above [23]–[26]. Although the initial Network Slicing visions and early concepts have been described already in 2015 [1], [2], it was only until October 2023 when the first, limited in scope commercial offering of Network Slicing has been announced [27][5]. A recent operator survey [25] reveals that similar, limited in scope commercial rollouts of this technology targeting selected types of Customers are planned by the Providers no sooner than in 2025 or later. It is because many Providers consider this technology to be still at its incubation stage and postpone its commercial adoption.

[3] In the report by Ericsson [177], 10 major Network Slicing business use cases are presented, with an extensive business analysis.

[4] Vertical industry is a group of companies that focus on a shared niche or specialized market spanning multiple industries. For example, Fintech companies facilitate financial services traditionally offered by banks. See [178] for further explanation of vertical vs horizontal industries.

[5] Deutsche Telekom established an NSI providing high throughput channel for transmitting signal from the TV broadcasting vehicles.

Indeed, several recent reports on the operational readiness of technologies that build up the Network Slicing landscape in various domains [28][29] state insufficient conceptual or technical maturity in several areas, which prohibits full-blown rollouts of Network Slicing with the envisioned features in their full extent. In particular, the broad study described in the author's major publication [30] indicates poor adoption of the NSIs/NSSIs allocation operations standardized by the 3rd Generation Partnership Project (3GPP) [31] and an insufficient coverage of aspects related to their implementation in the state-of-the-art. According to the author's firm conviction, **among the main deficiencies that limit widespread adoption of this technology there are the currently prevailing, not fully automated methods of NSIs/NSSIs design that prohibit their prompt, on-demand delivery**. Improvements in this area in full compliance to the Network Slicing provisioning and management standards in force [31], [32], including their supplementation with the missing design-time aspects[6], constitute the main subject of the research presented in this thesis, as described in the following Sections.

1.2. Motivation

In order to properly outline the motivation and research problem addressed in this thesis, the process of **NSIs/NSSIs allocation** from the Customers' and Providers' perspectives need to be described. For this purpose, the generic notions of resources provisioning and allocation are introduced below, in order to explain how they relate and differ from the specific semantics of NSI/NSSI allocation defined by the 3GPP standards[7] that are adhered to in this thesis.

In the context of Information Technology (IT) and telecommunications, **provisioning** is the process of *deploying and configuring hardware, software and/or communication medium resources* (i.e. CPU, memory, storage, network, but also Operating Systems (OS), applications and their settings, as well as bandwidth spectrum, Physical Resource Blocks (PRB)[8], etc.) *to ensure their operational readiness for the anticipated needs*. In turn, **allocation** is the process of *identifying and assigning concretized resources from the available pool to the requesting party in order to be utilized in a particular way*, e.g. for deploying a Virtual Machine (VM), running a containerized application or, in the case of NSI/NSSI, providing CSs with QoS guarantees. Thus, in order to allocate some resources, typically their prior provisioning needs to be performed. Usually, allocation requires specifying the precise "amount" or fraction of the quantifiable resources (e.g. count of CPU cores, amount of memory/storage, percentage of PRBs, etc.) and/or deciding on the exact configuration of the non-quantifiable resources (e.g. selection of concretized software package and its settings, choice and configuration of protocols, algorithms, scheduling priority, etc.) that shall be devoted to a given party.

In the view of these generic definitions, the particularities of the 3GPP-standardized **NSI/NSSI allocation semantics** can be compared and contrasted. Essentially, NSI/NSSI allocation is about providing an end-to-end telecommunication infrastructure (or its logical part) in which a CS that meets the QoS characteristics of the Service utilized by the Customers can be offered. According to the 3GPP standards[7], in order to satisfy the Customer's allocation request, the Provider may either reuse one of the existing NSIs/NSSIs (if it can still handle additional load), or may design and provision a new NSI/NSSI that will offer the expected characteristics. The latter choice, in turn, entails the design and provisioning of the necessary NSI/NSSI internal blocks (i.e. various types of Network Functions (NF), nested NSSIs, as well as their interconnecting Links[9]) or decision to reuse some of the existing ones, if they can still handle an additional load.

As such, NSI/NSSI allocation defined by 3GPP differs from the traditional perception of resources allocation, because NSI/NSSI is a distributed virtual entity composed of multiple internal building blocks which, in contrast to the "traditional" resources discussed above, **may or may not exist** at the moment of receiving the allocation request. Hence, NSI/NSSI allocation according to the standardized perception is a multiphase process, which may involve prior **complex design decision-making** and provisioning operations for deploying

[6] What is the "design-time" perspective of telco networks management is described in detail in the upcoming Subsections 2.2.3 and 2.3.4.

[7] See Clause 5.1.1 and 5.1.2 in 3GPP TS 28.531 [31] for the description of use cases that convey the semantics of the NSI/NSSI allocation operations.

[8] Physical Resource Blocks (PRB) are the basic units of radio medium division in the time and frequency domains. This term and its related concepts are intuitively explained in [179].

[9] A broad taxonomy of Network Slice/Subnet building blocks (referred by the author as *Slicing Entities*) is provided in Subsection 3.1.1.

and configuring the necessary building blocks, followed by deciding about allocating a fraction (or entirety) of their offered resources for satisfying the Customer's expectations.

A high-level organization of the NSI/NSSI allocation process performed according to the currently prevailing provisioning methods is depicted in Figure 1 and referred to in the paragraphs below.

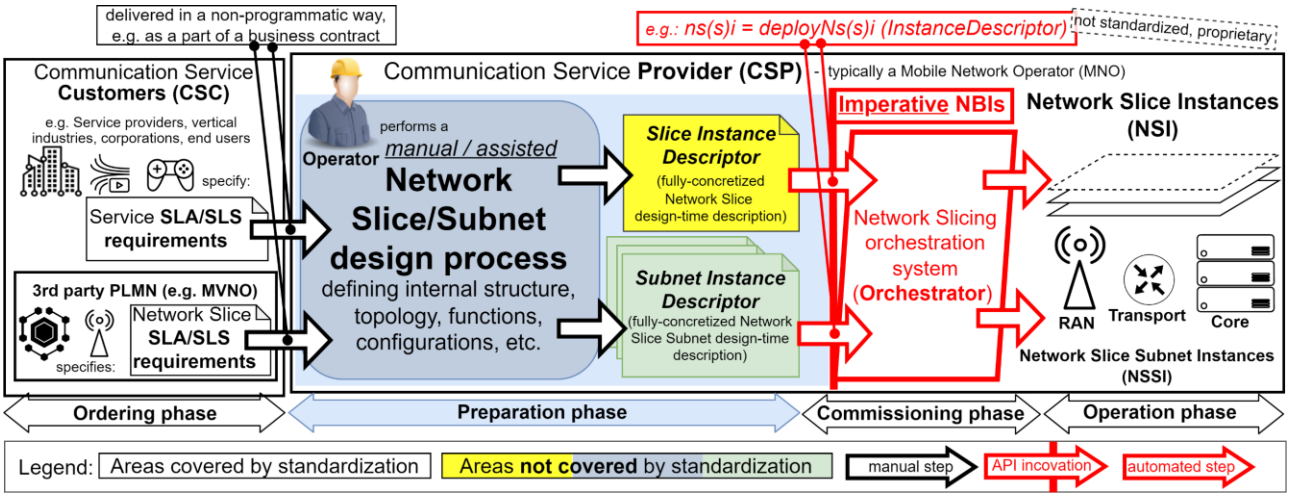


Figure 1. High-level organization of the Network Slice/Subnet design process according to the *imperative provisioning style*.

What also differentiates the allocation of NSIs/NSSIs from the allocation of "traditional" resources is the fact that the Customers who order allocation (depicted in the left side of Figure 1) **do not provide** any specification of the building blocks that shall build up the allocated NSI/NSSI, nor the "amount" or precise configuration of the resources they need. Instead, a given Customer provides to the Provider a set of high-level **Service-Level Agreement** (SLA) and/or a more technically-oriented **Service-Level Specification** (SLS)[10] **requirements** to be satisfied by the delivered NSI/NSSI perceived as a single, encapsulated unit. Such requirements are defined from the perspective of the QoS and high-level operational demands of the Service that will be accessible through a CS running on top of the ordered NSI/NSSI. As such, the Customer's defined SLA/SLS requirements may pertain to, among others: the coverage area, the latency threshold to be guaranteed, guaranteed mean throughput, the expected count of UEs that will utilize the Service and their geographical density, duration of the Service, etc.[11]

As an example, a Customer may be a vertical that plans deploying a SmartCity solution for improving coordination of the city traffic. For an uninterrupted and secure operation of such a strategically important Service, the vertical may order an Ultra-Reliable and Low-Latency Communication (URLLC) NSI to be allocated in the Provider's premises. In the allocation request, the vertical specifies the demanded coverage of the NSI and the uplink (UL) and/or downlink (DL) latency thresholds to be guaranteed. Or, the Customer may be a group of individual users who require a seamless communication channel (e.g. for smooth gaming experience, remote collaboration, teleconferencing, etc.). In such case, they may benefit from utilizing a dedicated NSI of type enhanced Mobile Broadband (eMBB) that provides guarantees of throughput and latency at the level required by their applications suite. As yet another example, the Customer may be a 3rd party Public Land Mobile Network (PLMN)[12], like a Mobile Virtual Network Operator (MVNO) that plans establishing its own NSIs of various types to be offered to its own Customers. Such PLMN may order the Provider to allocate NSSI(s) that will support the operation of its envisioned NSIs by offering demanded QoS characteristics in the areas covered by the Provider – in such use cases, the allocated NSSIs are utilized by the PLMN as building blocks of the NSIs it constructs.

As described above, allocation of NSI/NSSI from the Customer's perspective is envisioned to be **contract-based**. All that matters to the Customer is the fact that the allocated NSI/NSSI, perceived as a single end-to-

[10] The differences and relationship between SLA and SLS in the context of NSI/NSSI allocation is provided in Subsection 2.3.2.

[11] Further examples of SLA/SLS attributes are provided in multiple places of this dissertation. In particular, the upcoming Subsection 2.3.3 refers to the SLA/SLS attributes currently standardized by 3GPP in the `ServiceProfile`/`SliceProfile` data structures.

[12] According to 3GPP specifications [78][79], PLMN is an "entity which offers telecommunication services over an air interface providing mobile cellular services". An example of a PLMN is a particular MNO or MVNO. A given PLMN is identified by the Mobile Country Code (MCC) and Mobile Network Code (MNC). A list of such codes can be found on the following website: https://mcc-mnc.com/

end encapsulated unit, works according to the expectations expressed in the SLA/SLS requirements - the details of the NSI's/NSSI's internal technical realization and the resources that need to be utilized inside of the Provider's infrastructure **are not specified** by the Customer and, in principle, are not important from its perspective. Based on such a high-level input, **it is the Provider's duty** to **decide on reusing or newly designing** and provisioning a suitable NSI/NSSI in such a way that the entire, complex structure of its internal building blocks can be perceived as a unit that delivers the expected characteristics.

However, selection of a suitable NSI/NSSI or provisioning of a new one in the Provider's premises is not possible without thoughtful **interpretation** of the Customer's SLA/SLS requirements and **making a number of complex design decisions** on how to satisfy them with the resources available in the managed infrastructure. The Network Slice/Subnet **design process**, highlighted in Figure 1 with a blue background, aims at **elaborating a detailed, low-level technical specification** of an NSI/NSSI that will be capable of satisfying the Customer's SLA/SLS requirements. Designing each single NSI/NSSI is about deciding on a range of complex technical aspects that are prerequisites for its provisioning and operation in the Provider's infrastructure. Examples of such technical details to be determined during a design process include:

- deciding on the structure, topology and geolocation of NSI/NSSI's internals (NFs, Links, nested Subnets);
- deciding on the fraction of internal building blocks' resources to be devoted for supporting the desired operation of the currently designed NSI/NSSI (taking into account the already allocated ones);
- deciding on the operational configurations of all internals to guarantee QoS/SLA fulfillment;
- planning optimized deployment of the internal building blocks in the cloud environments, to save resources;
- defining QoS assurance policies at the level of individual internals and of the NSI/NSSI perceived as a whole;

as well as many other low-level aspects that need to be decided to guarantee satisfaction of Customer's demands by the designed NSI/NSSI. Such a technical specification, here and thereafter referred to as a Slice/Subnet *Instance Descriptor*[13], is then leveraged for executing the provisioning and allocation procedures in the Network Slicing management and orchestration system (referred to as an **Orchestrator**[14]) utilized by the Provider.

According to the traditional approach, referred to as ***imperative provisioning style***, the design process is conducted by a **human Operator** (Operator) – an expert, or a group of experts, managing the NSIs/NSSIs in the Provider's infrastructure by means of Orchestrator, as schematically depicted in Figure 1. In such a style, the Operator receives the specification of the high-level Customers' expectations expressed by means of the SLA/SLS statements and, based on the gained operational experience and domain knowledge, manually or in an assisted manner decides on all the enumerated aspects, crafting the specification of NSIs/NSSIs (*Instance Descriptors*) capable of fulfilling the Customers' requirements. During the design process, Operator takes into account the current and predicted state of resources in the managed infrastructure, the load and remaining capacity of the already running NFs and Links to be reused, the operational specifications of the ones possible to be newly spawned, remaining capacity of the deployment cloud environment, etc. Besides purely technical issues, the human Operator also needs to take into consideration the operational and business goals of the Provider, e.g. the cost of cloud infrastructure utilization and maximization of resources exploitation to increase profit, preferential treatment of selected Customers, etc.

Although the *imperative provisioning style* is a valid way of allocating NSIs/NSSIs, utilizing such style by the Providers exhibits several pain points that make it insufficient to fully satisfy the hopes placed in the Network Slicing technology mentioned earlier – in particular, regarding its flexibility, dynamicity, scale of orchestrated deployments and, as a consequence, its widespread availability. It is because making the design decisions by a human Operator is a highly **complex**, potentially **lengthy** and **error-prone** process, **complication of which increases** with the growing scale of operated environments and the growing count of orchestrated NSIs/NSSIs. Moreover, due to the **dynamicity** of the shared cloud deployment environments, fluctuating load of the already operated NSIs/NSSIs as well as their dynamic spawning and termination, the design decisions elaborated with large efforts of the Operator may turn out to be already outdated at the moment of their

[13] The term *Instance Descriptor* refers to a fully concretized design-time description of an entity – this term is elaborated in detail in Section 3.1.

[14] Although referred to in a singular form, Orchestrator denotes the entire stack of subsystems involved in the management and orchestration of Network Slicing across all layers of the administered 5G network.

finalization. Hence, fast handling of the incoming allocation requests is difficult to achieve, making the on-demand allocation of NSIs/NSSIs in the *imperative style*, even in small quantities, largely infeasible. The operational expenditures (OPEX) of the *imperative style* are also high, due to the **non-automated interactions with the Customers** and the need of human Operator's participation for conducting a **not fully automated design process**. All the factors enumerated above make the *imperative provisioning style* limiting the utilization and availability of the Network Slicing technology, at least in the nearest future, to a narrow group of "big players" Customers that can afford close collaboration with the Providers in setting up few large-scale and long-lived, statically operated NSIs like in the case of the majority of the ongoing trials [33], [34] and the mentioned first commercial rollout [5].

By taking into account the aspects discussed above, the main motivation of the research conducted by the author is to elaborate solutions improving and facilitating the NSIs/NSSIs allocation process. The elaborated results may contribute to the accelerated adoption of the Network Slicing technology, with the originally envisioned dynamic management, as well as its widespread utilization by a wide range of Customers. The author explores this challenge by tackling the topic of NSI/NSSI allocation process automation, which is an important issue not only from a practical and business point of view, but also from a cognitive one. In particular, the **problem of organizing and automating the NSI/NSSI design process** addressed in this thesis is specified in the following Section.

1.3. Problem statement

Widespread adoption of Network Slicing and its ubiquitous utilization requires bridging the gap between the Customers and the Providers, by providing both of these parties with **easier, more accessible** and **more efficient** methods of NSIs/NSSIs allocation. This need calls out for employing more effective orchestration techniques by the Providers that assure **full automation** of the allocation process and **simple** way of NSIs/NSSIs ordering by the Customers. Such improvements should allow for dynamic, on-demand delivery of NSIs/NSSIs in massive quantities, including small-scale, short-lived ones delivered to the individual Customers, enabling the realization of the original Network Slicing visions in their full extent.

To achieve these goals, a major conceptual shift in Network Slicing provisioning is promoted by the recently emerging trend of ***declarative style*** telco networks management and orchestration. According to the *declarative style* principles, the Orchestrator shall expose an Application Programming Interface (API) that enables its users to declare **what** final effect shall be achieved (e.g. a running NSI and its constituting NSSIs that satisfy a given set of QoS requirements), **without the need of specifying how** to achieve it on the technical level – **it is the Orchestrator's responsibility, not the human Operator's, to automatically devise a way to satisfy the Customer's request** and apply it in the managed environment.

The principles of the *declarative style* have been recognized by the standardization bodies in the Network Slicing domain on the business [35] and technical levels [31], [32] already for some time[15]. Among the most prominent examples[16], which actually triggered the author's research presented in this thesis, the *declarative provisioning style* is enforced by the signatures of the **3GPP-standardized `allocateNsi` and `allocateNssi` operations** [31] that shall be exposed in the Northbound Interfaces (NBI) of the standards-compliant Orchestrators utilized by the telco Providers [17]. The input to these operations are single **`ServiceProfile`/`SliceProfile`** data structures [32], which only convey the high-level and, in many cases, abstract **SLS**[10] **requirements** to be satisfied by the allocated NSI/NSSI, without any low-level details on how they should be technically realized in the Provider's infrastructure. As such, it can be stated that the *declarative style* of NSIs/NSSIs provisioning at the technical level is directly inspired by the way of their

[15] See the upcoming Section 1.6 for the presentation of the author's research timeline against the backdrop of Network Slicing standardization.

[16] Subsection 2.5.3 discusses other examples of *declarative style provisioning* approaches considered in the state-of-the-art research.

[17] Until version v18.1.0 of the 3GPP TS 28.531 specification [180] released in March 2023, only `allocateNsi`/`allocateNssi` operations have been standardized and they still remain specified as synchronous, blocking operations. However, the author is convinced that all the complicated processing behind these operations will rarely be possible to be performed within timeframes considered reasonable for synchronous calls (e.g. within less than 60s). This prediction has been confirmed by the performance evaluation results described in Section 6.4. Hence, redefinition of these operations' specification in the standards as asynchronous has been suggested by the author in [30]. It also needs to noted that during the works on this dissertation, an alternative, asynchronous way of allocation has been introduced in June 2023 (3GPP TS 28.531 v18.2.0 [31]), performed by means of `NetworkSliceController`, `NetworkSliceSubnetController` models newly specified in 3GPP TS 28.541 v18.4.1 [32]. Nevertheless, in this thesis and in the implemented PoC, full alignment to the `allocateNsi`/`allocateNssi` specification is pursued.

ordering at the business level (described in the previous Section) – contract-based, where the contract is defined by the `ServiceProfile`/`SliceProfile` data structures consisting of a set of SLS statements. The schematic, high-level organization of NSI/NSSI allocation process according to the *declarative provisioning style* standardized by 3GPP is depicted in Figure 2.

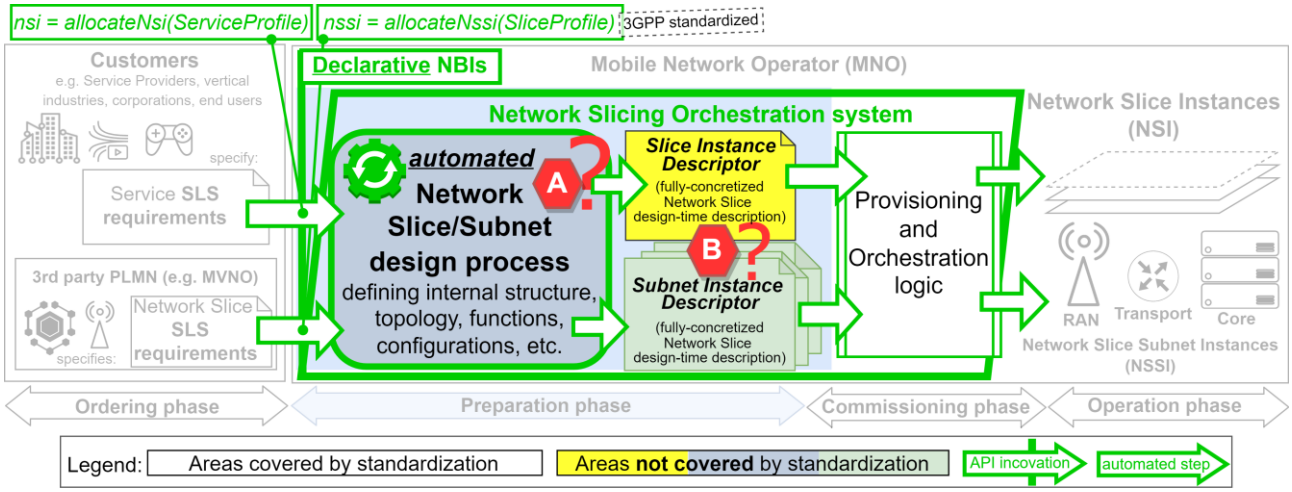


Figure 2. High-level organization of the Network Slice/Subnet design process according to the *declarative provisioning style*.

Importantly, according to the *declarative provisioning style* principles, the Customers may be enabled by the Provider to trigger NSI/NSSI allocation **themselves**, **by directly interacting with the Orchestrator in a programmatic manner via its exposed NBI** - this is in clear contrast to the *imperative provisioning style*, where the interactions with the Orchestrator are restricted to expert Operators only. All the enumerated characteristics of the *declarative provisioning style* promise convenient and easy NSI/NSSI allocation from the Customers' perspective, as it **does not require any low-level telco domain knowledge**. Moreover, such style offers high flexibility of NSIs/NSSIs allocation, as the Customers can encode the interactions with the Provider's Orchestrator via an API, just like in case of a programming library. Thus, the *declarative style* of NSIs/NSSIs allocation is perfectly aligned with the core characteristic of cloud computing [36], namely the **self-service**, which lets Customers obtain and remove cloud services by themselves, without requiring the assistance of an Operator or IT staff member. As such, it can be stated that Network Slicing complements the already achievable capabilities of the cloud environments (i.e. virtualized compute, storage, localized networking, etc.) with the crucial capability that was missing in the cloud environments until now – easy and prompt establishment of virtualized, wide-area wireless "last mile" connectivity with QoS guarantees for seamless interactions of the Customers with the Services.

However, practical realization of the *declarative provisioning style* is challenging for the Providers due to a number of technical and organizational factors. First of all, supporting *declarative* NBIs requires the Providers' Orchestrator to be equipped with advanced functionalities, capable of leveraging **domain knowledge** for appropriate **interpretation** of the SLS statements and **inferring** the technical specification of an NSI/NSSI that will satisfy the Customer's QoS/QoE expectations. Secondly, to support the key Slicing virtues, namely shareability and maximization of resources exploitation, the NSIs/NSSIs delivered to the Customers shall either be **reused** or be designed from a mixture of already existing internal building blocks (if additional load may be still handled by them) and the ones to be newly spawned. Determining which of such elements to use for satisfying the Customers' expectations requires continuous tracking and planning of their SLS fulfillment, supervision of their QoS satisfaction capabilities as well as embedding the design intelligence into the Orchestrator. Thirdly, the design decisions need to take into account the scale and **dynamicity** of the already existing, simultaneously running NSIs/NSSIs, which not only share their building blocks and the underlying resources, but themselves may be shared by the higher-level entities[18]. Finally, the Orchestrators supporting declarative NBIs need to be prepared for prompt and dynamic handling of NSI/NSSI allocation requests that are triggered by the Customers **spontaneously**, out of Provider's control. Hence, timely

[18] According to standardized models and lifecycles, NSSI may be shared by multiple NSIs, and NSI may be shared by multiple CSIs, see Subsection 2.3.1.

handling of the allocation requests enabling on-demand NSIs/NSSIs delivery **excludes any manual steps** by a human Operator.

The aforementioned challenges are infeasible to be embraced without proposing the means **to automate the entire allocation process in an end-to-end manner** – from the reception of the SLA/SLS requirements from the Customers, through the **design of an NSI/NSSI capable of fulfilling them**, up to its provisioning in the distributed cloud environments. While the provisioning procedures executed based on concretized *Instance Descriptors* are already largely automated in the currently available Orchestrators, the author concentrates on the issues of **automating the design process**, which at the time of the conducted research remained an area not thoroughly covered in the state of the art. The author claims so, because the standards specifications, besides dictating the signatures of the declarative `allocateNsi`/`allocateNssi` operations [31], **do not provide any guidelines** that would allow for addressing the aforementioned design-time challenges. Also outside of the standardization, among the state-of-the-art research, there is lack of comprehensive solutions that would address the enumerated design-time issues in a generic and holistic manner (see related work in Section 2.5).

In particular, the standards do not provide any specification of Network Slicing **design-time models**, making it unclear what information should be contained in a technical description of an NSI/NSSI before its deployment, as well as in what form should such a description be conveyed during the design process and after its completion (Figure 2, badge A). Furthermore, there is no specification of any **design-time procedures** that would transform the abstract SLA/SLS statements into a concretized design-time description (*Instance Descriptors*) of an NSI/NSSI capable of fulfilling them (Figure 2, badge B). Due to these gaps, it is unclear **how to organize a Slice/Subnet design process** that would be aligned with the *declarative provisioning* operations. All the aforementioned factors call out for comprehensive research in the Network Slicing design-time area and justify **focusing this dissertation on the organization of an automated design process**, including **devising the standards-aligned models and methods** allowing its enablement.

1.4. Thesis statements and dissertation goals

Based on the motivations and problem statement described in the previous Sections, the following thesis is formulated by the author:

With the synergy of modeling the Operator's and Provider's domain, operational and business knowledge, uniform modeling of the Network Slice/Subnet building blocks and the structuralization of the design activities to leverage the modeled knowledge, it is possible to define a coherent and generic conceptual framework of a design process that enables construction of multivariant and extendable systems performing automated transition from high-level SLS requirements to a technical specification of an NSI/NSSI capable of meeting them.

The ultimate goal of this dissertation is to verify the thesis statements by:

1. formulating conceptual foundations of a generic Network Slice/Subnet design process automation solution, including the proposition of models and methods that combined together enable construction of customizable, multivariant systems automating the processes taking place in the design phase;
2. confirming the feasibility of practical implementation of the elaborated concepts by developing a Proof-of-Concept (PoC) solution that supports selected, most essential variants of the generic solution;
3. evaluating the functional and non-functional aspects of the elaborated solution in order to assess its applicability to support the declarative, on-demand NSI/NSSI allocation in various scales, which is anticipated to be performed in the Network Slicing Orchestrators in the nearest future.

1.5. Contributions

In light of the thesis statements, the author's main contributions elaborated in this dissertation can be generalized as pertaining to the **representation of expert knowledge** in **appropriately defined models** and **establishing structured way of their processing**, which are leveraged for automating complex procedures and decision-making involved in the Network Slice/Subnet design process. In particular, the thesis is verified

by exploring the following key concepts contributed by the author and elaborated in detail in the course of this dissertation:

A. **uniform modeling** of the Network Slice/Subnet building blocks as encapsulated units with exposed **Capabilities** that correspond to the SLS requirements posed against them;
B. **modeling of** Operator's and Provider's domain, operational and business **knowledge** that enables its easy algorithmic processing;
C. **decomposition** of the complex design process and its **fine-grained structuralization** into a set of coherent, single responsibility design activities with clear input/output and complementary roles that leverage the captured domain, operational and business knowledge.

With more granularity, the range of important contributions elaborated on the way towards verifying the thesis is enumerated in the list below:

1. broad analysis of the evolution and the current **standardization landscape** related to 5G and Network Slicing management, identifying the gaps regarding the Network Slicing design-time aspects;
2. specification of **functional and non-functional requirements** to be met by a Network Slicing design automation solution that supports declarative, hands-free and dynamic allocation of NSIs/NSSIs;
3. **critical analysis of the state-of-the-art research** related to the Network Slicing design-time aspects, confronting the existing solutions against the defined requirements and innovative approach employed in the author's work;
4. formulation of clear **definitions** for the fundamental design-time aspects that are currently left without common understanding in the state-of-the-art:
    a. definition of the Network Slice/Subnet design process;
    b. definition of the expected Network Slice/Subnet design outcomes;
5. proposition of a **broad set of constructs that enable effective modeling** of the design process inputs and outputs, with a particular focus on the domain and operational knowledge, capturing and representation of which in appropriate models is essential for conducting the design process in an automated manner. The most important of the proposed modeling constructs and systematizations include:
    a. modeling construct of a ***Slicing Entity*** that serves as a single, uniform abstraction over a large heterogeneity of building blocks out of which a Network Slice/Subnet may be designed;
    b. comprehensive **taxonomy** of *Slicing Entities*, taking into account criteria important in the design process: lifecycle stage, concretization and abstraction levels, class, type and utilized models;
    c. modeling constructs of *Slicing Entity* ***Capabilities*** and ***Configuration Profiles***, introduction and exploitation of which is essential for the organization of an automated design process that is based solely on SLS requirements delivered by the Customer;
    d. **deep analysis** of standardized **management models**, including identification and classification of information items that need to be contained in a Slice/Subnet *Instance Descriptor*, focusing on determining methods of automated delivery of values for the identified items;
    e. structure of **standards-compliant Slice/Subnet *Instance Descriptors*** that are based on the aforementioned modeling constructs, conducted classifications and systematizations;
    f. generic and versatile **model of a *Capability/Requirement attribute*** that conveys the knowledge regarding methods of its automated processing in various design activities;
    g. generic **models** enabling **flexible *Requirements* decomposition** and ***Capabilities* aggregation**;
6. systematization of a holistic, **standards-aligned framework of Network Slice/Subnet design process automation**, including:
    a. decomposition of the design process into a set of fine-grained, coherent and generic, single-responsibility design activities with clear input/output and complementary roles;
    b. deep analysis of each design activity for devising the methods of their automation;
    c. clear roadmap for a gradual introduction of automation in the design-time phase, based on incremental, backward-compatible *Automation Scopes*;

7. **PoC implementation** of the selected *Automation Scope* within a sample Orchestrator, proving the feasibility of the elaborated concepts and showcasing their practical realization;
8. **experimental evaluation** of the implemented solution in conditions closely mimicking real Network Slicing operational scenarios, including description of the conducted functional and performance tests, as well as the analysis and interpretation of the obtained results;
9. discussion of the potential for **industrial applicability** of the thesis results.

In the course of this dissertation, it is shown how the enumerated contributions serve as building blocks for organizing an automated Network Slice/Subnet design process which, in turn, allows for verifying the thesis.

1.6. Scope

The purpose of this Section is to further clarify the area of focus of this dissertation by specifying the boundaries of the author's research. To this end, the covered area of interest is narrowed by considering various perspectives discussed in the following paragraphs.

Firstly, it shall be noted that in general, the Network Slicing idea may be applied to various types of telecommunication and computer networks, including mobile or fixed-line networks, Local Area Networks (LAN), Wide Area Networks (WAN), etc. This thesis, however, considers **the context of the 5G mobile telecommunication networks only**, without taking into account the specifics of other network types.

Secondly, the Network Slicing technology manifests its presence in virtually every layer of a 5G network, not only at the technical, infrastructural and architectural, but also at the organizational and business levels. In this view, the contributions of this thesis pertain to the technical, architectural and organizational aspects, **focusing on the design-time issues** within the Network Slicing **management layer** (see Figure 3 below). Due to the fact that the functioning of this layer, and of the design process in particular, is influenced by the neighboring business and 5G infrastructural layers, their selected aspects are considered in this thesis as well. However, it shall be noted that the low-level technical aspects of the telco infrastructure functioning (e.g. the operation of the particular types of 5G NFs, links, call processing protocols, radio medium utilization techniques, etc.), although touched upon in this thesis, are not in scope of this work.

Thirdly, management of the telco equipment is traditionally considered from perspectives collectively referred to as FCAPS, which stands for: Faults, Configuration, Accounting, Performance and Security. With respect to FCAPS it shall be underlined that the responsibility of the design process is solely to produce decisions that will configure the **control-plane** of the building blocks comprising the designed NSI/NSSI, preparing them for handling the input SLS requirements specified by the Customers. Hence, in this thesis, the focus is on the Configuration Management (CM) perspective, also taking into account the Fault and Performance Management (FM/PM) metrics from the infrastructure operation as an important input influencing the decisions in a design process. The Accounting and Security perspectives, however, are out of scope of this thesis.

Fourthly, taking into account the NSI/NSSI lifecycle[19], the scope of this thesis concentrates on the design-time issues that occur in the **preparation phase** of the end-to-end lifecycle, as it has been already depicted in Figure 1 and Figure 2. But, due to their relevancy for the design process, selected aspects of the ordering, commissioning and operation phases are discussed as well. It needs to be underlined, though, that the operational functioning of the designed NSIs/NSSIs and their internals, and in particular whether they do deliver the expected level of SLS satisfaction for the user-plane traffic with the applied control-plane configuration decided during the design process, is a problem belonging to the SLS assurance research area, which is broadly addressed in the state-of-the-art (e.g. [5], [6], [37]–[42]), but not targeted in this thesis.

Fifthly, as outlined in the previous Sections, the major contributions targeted for elaboration in this thesis pertain to the modeling, exposition and processing of various pieces of domain and operational knowledge that is needed for conducting the design process in an automated manner. However, **the particular specimens of the domain, operational or business knowledge are not in themselves** among the contributions of this thesis and their detailed elaboration is beyond the scope of this work. Thus, the multiple

[19] Subsection 2.3.1 contains a detailed explanation of phases comprising the standardized NSI/NSSI lifecycle.

examples of such knowledge presented at various stages of this dissertation (including SLS attributes' semantics and values, configuration settings, decomposition and aggregation rules, topologies, etc.) serve merely as examples and should not be perceived as an operational reference. Delivery of this knowledge is the responsibility of the equipment Vendors, Providers and Operators managing the telco infrastructure.

Last, but not least, this thesis puts **standards-compliance** as an important determinant of the genericity, coherency and potential for broad exploitation of the research outcomes, including their potential industrial applicability. In this respect, it needs to be noted that since the formulation of the 5G Network Slicing concept in 2016 [1], there have been a number of stakeholders involved in shaping its development, from various technological and business perspectives. These stakeholders can be categorized into several groups, including: vertical industry organizations representing the main groups of Customers, telco industry organizations, Standards Development Organizations (SDO), open-source projects and research initiatives, telco Providers, as well as Vendors developing commercial products. The scope of work of these organizations and the relationship between them is schematically depicted in Figure 3.

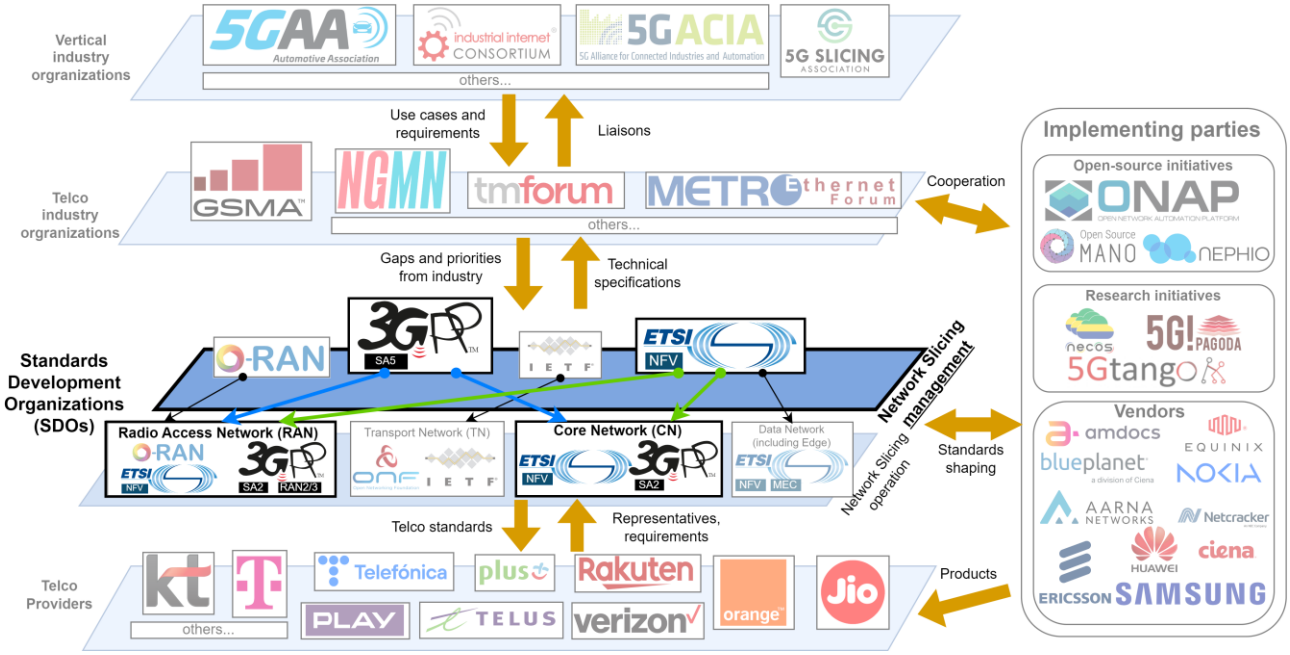


Figure 3. Landscape of organizations involved in shaping the Network Slicing technology and relationships between them. The scope of this dissertation is related to the scope of management aspects standardization by 3GPP and ETSI. Figure based on [43].

Within such a complicated landscape, the key responsibility of the SDOs is to develop technical specifications that satisfy the operational and business objectives of the telco Providers, Vendors and Customers. However, lack of harmonization between the efforts of the enumerated stakeholders, in particular their frequently disjoint or overlapping scopes and contradicting priorities, resulted in a hyper-fragmented ecosystem in which there is an abundance of technological and business viewpoints, leading to a proliferation of open issues and gaps that need to be bridged[20]. The abundance of stakeholders and the lack of their coordinated work is frequently considered among the major factors slowing down the commercial rollout of this technology.

Moreover, it needs to be noted that the involvement of the aforementioned stakeholders has been provided **at various timeframes**, which overlapped with the author's research activities spanning from mid-2019 to mid-2024, as schematically depicted in Figure 4 below. Within this time, the author produced several research items in the Network Slicing area, including a major research paper [30] and a number of inventions for patenting, out of which one has been already granted [44] and the remaining eight are pending in the patent offices around the world [45]–[57]. Due to the mentioned overlapping, the **timing of SDOs deliverables** and their advancement at the particular moments in time had a direct influence on the scope of research presented in this thesis. More specifically, as mentioned in Section 1.3, the primary trigger of the author's research was the specification of the `allocateNsi`/`allocateNssi` operations and of the

[20] In this thesis, the author identifies and fills-in the gaps related to the Network Slicing design-time aspects, pointed out in Sections 2.2 and 2.3.

`ServiceProfile`/`SliceProfile` models within the 3GPP Network Slicing provisioning standards, which were introduced in the 28.531 and 28.541[21] specifications in mid-2018 [58][59] and remain the part of these standards until now [31], [32]. This trigger positioned the author's research and the elaborated contributions mainly **within the scope of 3GPP Network Slicing provisioning standardization** and the closely related, already well-annealed at that time, standards of the European Telecommunications Standards Institute (ETSI) of the Network Functions Virtualization Management and Orchestration (**NFV-MANO) framework**.

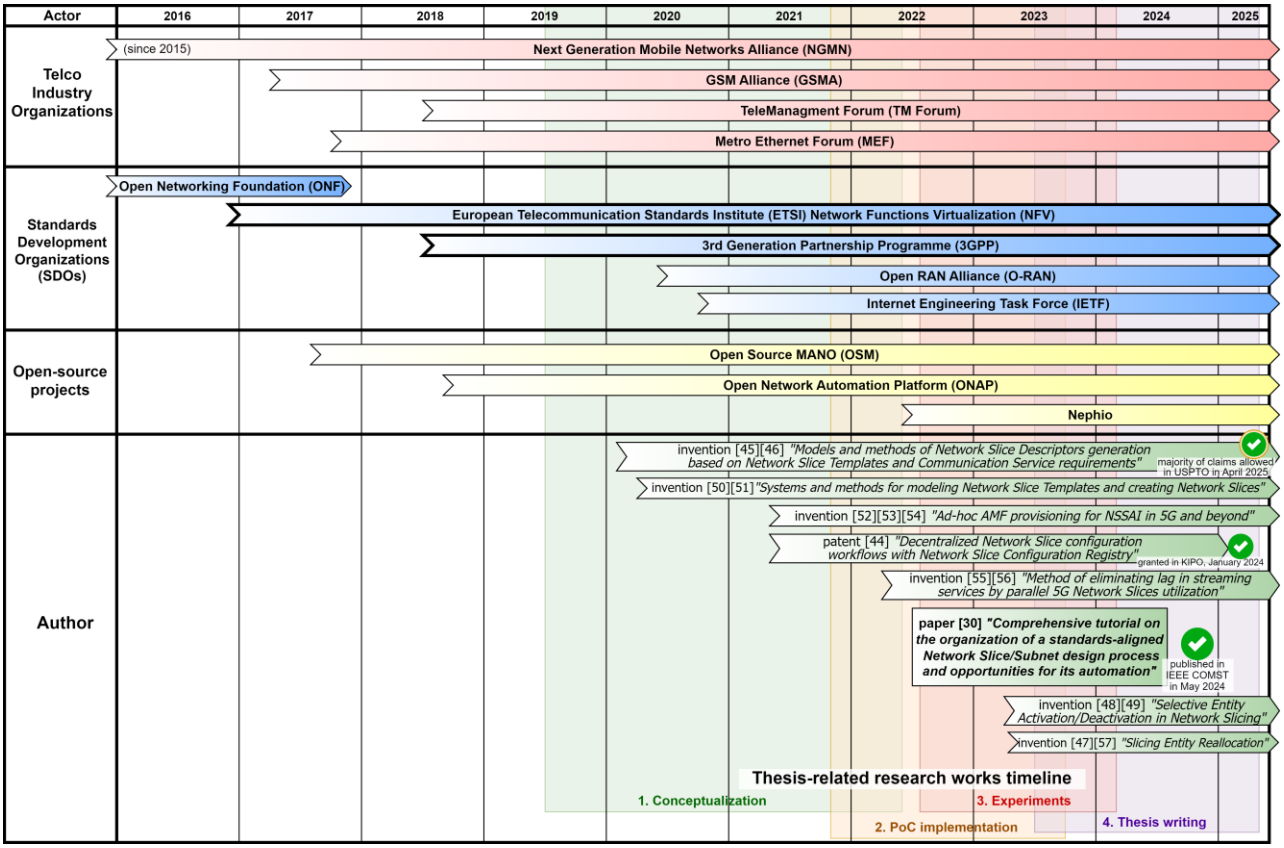


Figure 4. The timeline of the author's research against the backdrop of Network Slicing technology development in the main industrial, standardization and open-source communities. Figure based on [43].

However, while 3GPP covers a vast majority of the 5G technological landscape, including the management and provisioning aspects of end-to-end Slices, Subnets, down to the level of individual 5G NFs and Links in the Core and Radio Access Network (RAN) domains, the domains of Transport Networks (TN) and Data Networks (DN) are out of scope of 3GPP standardization, as depicted in Figure 4. According to the SDOs' consensus, these areas shall be covered by the Internet Engineering Task Force (IETF). However, IETF started working on Network Slicing specifications at a relatively late stage, only beginning in mid-2020 with several poorly coordinated drafts that have been under development in various incarnations, e.g. [60]–[63]. Eventually, around mid-2022, these spread efforts started to take shape as merged "Internet-Drafts" that are continuously developed until today[22]. Nevertheless, due to the immaturity of these works at the timeframes of the author's research, **designing Subnets within the TN domain**, selected aspects of which are touched upon in this thesis, **is not within its scope**. Adoption of the elaborated concepts for designing TN Network Slices, by taking into account the deliverables of the continuously evolving IETF specifications, is considered as a future work.

Finally, it needs to be mentioned that while the utilization of Artificial Intelligence (AI) techniques for tackling the Network Slicing design-time issues is not in scope of this dissertation, the emerging research initiatives attempting such approach are commented in Subsection 7.4, in the context of future works.

[21] Organization of the 3GPP and ETSI standardization processes and kinds of specifications produced by them are explained in the Appendix.

[22] Recently, in mid-2024, the IETF's Internet Draft describing the conceptual framework for managing so-called "IETF Network Slices" reached the status of an RFC [181]. However, the corresponding technical specification defining the necessary APIs and data models [182] is still in a drafting stage. Another important Internet Draft, related to the application of "IETF Network Slices" within the 3GPP end-to-end Slices, kicked-off in mid-2023 [183].

1.7. Structure

The reminder of this dissertation is organized as follows. Chapter 2 contains extensive introductory information that provides the necessary context to properly situate and understand the aspects discussed in the rest of the work. Among others, this Chapter analyzes and explains the standardized concepts, definitions, models, roles, procedures, offering modes, systems and APIs related to the management of 5G networks in general and the Network Slices and Subnets in particular. Importantly, the conducted analysis identifies critical standardization gaps related to the Network Slicing design-time aspects, establishes the requirements that the sought solution must meet, and compares the author's research with the other state-of-the-art initiatives.

Chapters 3 and 4 are devoted to providing the necessary conceptual foundations based on which the automation of the Network Slice/Subnet design process is tackled in this dissertation. More specifically, Chapter 3 describes the basic concepts, taxonomies and systematizations of the design-time aspects that are currently not defined in standardization nor in the state-of-the-art research, focusing on a deep study and modeling of the inputs and expected outputs of the design process. In particular, the multiple modeling constructs introduced in this Chapter allow for conveying the necessary operational and domain knowledge, playing a key role in enabling the automation of the design process. Chapter 4 continues the elaboration of conceptual foundations, by focusing on the analysis of the course of the design process in a generic case. As a result, this complex process is decomposed into a set of fine-grained, coherent, single-responsibility design activities that are analyzed individually for identifying the methods of their automation, leveraging the expert knowledge contained in the elaborated models. Combined together, all the concepts introduced in Chapters 3 and 4 formulate a conceptual Network Slice/Subnet design process automation framework that can be leveraged for the realization of design automation solutions in various scopes.

In turn, Chapters 5 and 6 describe the practical verification of the concepts elaborated in the earlier Chapters. In particular, Chapter 5 proves the feasibility of the elaborated concepts by showcasing how a selected variant of the proposed framework has been embodied in practice. The implemented PoC solution augments a sample Orchestrator, which previously followed the *imperative provisioning style* and required conducting the design process in a manual manner, with the feature of performing a fully automated design process. As a result, the augmented Orchestrator supports a purely *declarative style* of provisioning in full compliance with the declarative allocation APIs standardized by 3GPP. In turn, Chapter 6 is devoted to an experimental evaluation of the implemented PoC solution, testing it from functional and performance perspectives. The extensiveness of the conducted tests and the analysis of the obtained results not only deliver measurable indicators supporting thesis verification, but also provide a broad insight into the functioning of the implemented solution. Finally, Chapter 7 concludes this dissertation by validating the thesis statements, discussing the potential for industrial exploitation of the thesis outcomes and outlining the future works.

Chapter 2

Background

In this Chapter, the context of the conducted research is outlined, by referring to the state of the art and standardization of 5G in general and the aspects of Network Slicing management in particular. Since standards-alignment of the concepts elaborated in this thesis is among its core values, comprehensive explanation of the most relevant of the standardized telco networks management concepts, models, systems and procedures, but also the corresponding nomenclature and abbreviations is essential for proper understanding of the content provided in the upcoming Chapters. Hence, the Chapter's content is organized as follows.

Firstly, basic information about the architecture and elements comprising a 5G network is provided. Then, a systematic review of the 5G and Network Slicing management aspects' standardization is conducted in a gap analysis manner. The review is performed from the perspective of NSI/NSSI design process automation that is aimed to be achieved, allowing for clear indication of the gaps that are the target of the author's research. Finally, the presented analysis also allows for clear presentation of related work, by narrowing down the literature review to research items that tackle similar aspects.

2.1. 5G basics

Diving into the details of Network Slicing is impossible without introducing the basic technological aspects of 5G. In this Section, the most important features of 5G in the layers relevant for Network Slicing management and orchestration are introduced.

2.1.1. Requirements and expectations

The process of defining 5G technological stack has been driven by the specification of requirements defined within the International Mobile Telecommunications-2020 (IMT-2020) standardization initiative [64], steered by a sector of the International Telecommunication Union (ITU) responsible for Radiocommunication (ITU-R). In a similar way, requirements for the 3G, 4G and 6G technologies have been specified in the IMT-2000 [65], IMT-Advanced [66] and the ongoing works on the IMT-2030 [67] standards, respectively. The specified requirements are presented in a compact manner by so-called "spider charts" depicted in Figure 5.

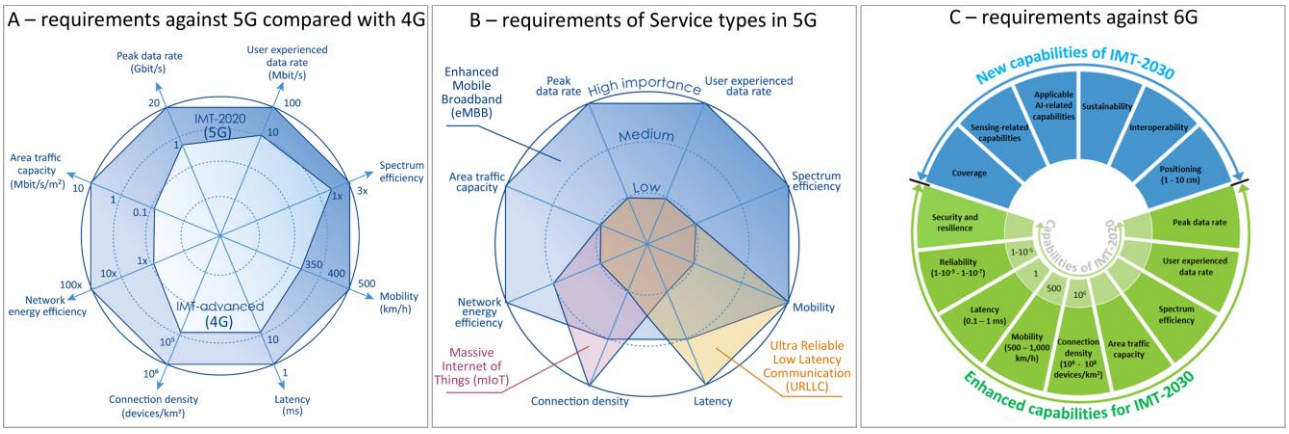


Figure 5. Requirements against the 4G, 5G and 6G networks and of the various Services they are expected to support.

As depicted in Figure 5.A, comparing to the preceding 4G networks, 5G is expected to improve on all of the considered dimensions, including increased data rates reaching 20 Gbit/s, increased spectrum and energy

efficiency, increased traffic capacity and connection density, as well as lowering down the achievable latency to the value of 1ms. What is important to note is that unlike 4G, which provides the "one-size-fits-all" connectivity with the characteristics presented in Figure 5.A, the requirements to be met by 5G with all their stringent values cannot be delivered at once by a single, common network. Instead, characteristics demanded by various types of Services, which are schematically depicted in Figure 5.B, can only be supported by 5G networks that are configured in a particular way. In fact, the need for supporting the presented, divergent requirements simultaneously by means of a single underlying network infrastructure was among the main reasons for the establishment of the Network Slicing technology.

As an example, a Slice of type eMBB should offer characteristics allowing smooth operation of general-purpose Services that do not pose stringent requirements related to connection density and latency. A Slice supporting URLLC-typed Services should deliver uninterrupted, low-latency communication platform that is not expected to support massive count of users and does not need to offer peak data rates. In turn, a Slice of type mIoT should be capable of supporting Services with large number of densely operating UEs that do not require high data rates or low latency communication, but should be supported by the network in an energy-efficient manner. While only three types of Services to be supported by the 5G networks/Slices have been presented in Figure 5.B, there are currently 6 standardized types defined in Clause 5.15.2.2 of [68], identified by the Slice/Service Type (SST) values listed in Table 1. The expected characteristics of a Slice with a given SST value are specified in [69]. Besides the standardized ones, Slices supporting customized Service types may be offered by the Providers with SST values out of the presented range.

Table 1. Meaning of the 3GPP-standardized Slice/Service Type (SST) values.

Slice/Service type	SST value	Full name
eMBB	1	enhanced Mobile Broadband
URLLC	2	Ultra-Reliable Low Latency Communications
mIoT	3	massive Internet of Things
V2X	4	Vehicle-to-Everything
HMTC	5	High-Performance Machine-Type Communications
HDLLC	6	High Data rate and Low Latency Communications

Requirements against 6G are even more stringent, again outranging the preceding generations in all of the considered dimensions. Moreover, requirements against new capabilities to be offered by the network are posed, which is schematically depicted in Figure 5.C. Similarly to 5G, it is expected that offering all the demanding characteristics within a common network with general configuration would not be possible, making Network Slicing a fundamental technology in the ongoing research towards 6G.

2.1.2. Key architectural principles of 5G

The architecture of 5G represents a significant evolution from previous generations, designed not only to meet the increasingly demanding requirements of connectivity, speed, and latency described above, but also to allow the introduction of advanced features in the management and orchestration layer that allow more efficient utilization of the network infrastructure by the Providers. In particular, Network Slicing is possible to be introduced in 5G thanks to several architectural pillars, including: **Network Functions Virtualization** (NFV), **Service-Based Architecture** (SBA) and **Control User Plane Separation** (CUPS). These architectural aspects are briefly introduced below, providing the necessary context for the discussions in the further course of this dissertation, including presentation of the types of 5G Network Functions (NF) and their responsibilities, which are discussed in the next Subsections.

Unlike in 4G, where NFs typically constituted vertically integrated boxes tightly bound to specific hardware devices, almost all 5G NFs are expected to be virtualized. NFV plays a pivotal role in the 5G architecture by decoupling of hardware and software functions to enable more flexible and efficient resources management. Through virtualization, the NFs that were traditionally delivered as vertically-integrated boxes referred to as Physical Network Functions (PNF) become **Virtualized Network Functions** (VNF) implemented as software applications running in virtual machines (VM) or Operating System-level containers[23]. Typically, VNFs are developed as cloud-native applications built according to the microservices architectural principles [70], making them deployable in both private, Provider-owned cloud infrastructure as well as in public clouds. This

[23] To distinguish the container-based VNFs, the notion of a CNF (Containerized Network Function) is frequently utilized in the telco community. It is, however, not a term that is standardized, because only the VNF notion is utilized in the ETSI specifications.

allows the traditionally closed telco ecosystem to benefit from the capabilities of the state-of-the-art cloud infrastructures and advanced software engineering tools that enable flexible and scalable operation of VNFs, as well as their seamless deployment in open-source based virtualization solutions, like OpenStack or Kubernetes clusters [24]. Besides the abovementioned virtues, virtualization also facilitates dynamic deployment of new NFs and modification of existing ones, making NFV one of the key pillars for Network Slicing realization.

SBA specified in [68], [71] is another cornerstone of the 5G architecture that introduces a modular and flexible framework for the interworking of various types of 5G NFs. To follow the SBA principles, 5G NFs need to be implemented as discrete services that can be accessed through APIs standardized in the 3GPP specifications, e.g. [72], [73]. Thanks to such approach, SBA promotes interoperability and flexibility, allowing network components to interact seamlessly regardless of the underlying technology or Vendor. This openness accelerates innovation and facilitates the integration of third-party services, contrasting with the monolithic architectures of previous generations, where functions were tightly coupled and difficult to modify boxes delivered by the particular Vendors. In the SBA framework, each NF is implemented as a microservice that can be independently developed, deployed, and scaled. The modularity of SBA supports creation of tailored service instances, optimizing resources utilization and improving overall network efficiency. Additionally, SBA enhances network resilience and reliability, as services can be dynamically reconfigured or replaced without impacting the entire network.

CUPS is another fundamental design principle in the 5G architecture that decouples the NF functionalities into distinct NFs that handle control plane and the user plane separately. The control plane NFs handle signaling, mobility management, and session management, while the user plane NFs are responsible for data forwarding and traffic management. This separation enhances the scalability and flexibility of the telco network by allowing independent scaling and evolution of each plane. In the context of Network Slicing, CUPS enables easy deployment of Slice-dedicated NFs as well as simplified reconfiguration of NFs that are shared by multiple Slices. CUPS also facilitates the deployment of VNFs in distributed DCs, including so-called “Edge” [74], [75] sites that offer computational resources closer to the end users.

The combination of the 5G architectural foundations described above provide the necessary means for dynamic and flexible management of virtualized resources, constituting important enablers for realizing the visions of the Network Slicing technology.

2.1.3. 5G functional architecture

By taking into account the architectural principles introduced in the previous Subsection, the functional architecture of NFs and Links building up the 5G landscape can be introduced, which is schematically depicted in Figure 6 below. The presented architectural view consists of five major parts: the Users space, the Access Network (AN) domain, Core Network (CN, Core) domain, the Data Network (DN) domain as well as the Transport Networks (TN) that are connecting these domains in various arrangements. Starting at the lower left of the diagram, there is the UE[2]. UE can be connected to various ANs[25], with the most important variant for the mobile access being the Radio Access Network (RAN). RAN connects the UEs to the Services in the DN by means of NFs comprising the CN domain. In the following Subsections, the most important types of NFs and Links in the RAN and CN domains are presented, including enumeration of their functionalities and responsibilities.

[24] OpenStack (https://www.openstack.org/) and Kubernets (https://kubernetes.io/) are among the most popular cloud computing platforms supporting orchestration of VM-based or OS containerization-based virtualized workloads, respectively.

[25] There are other non-3GPP accesses like WiFi or Satellite communication.

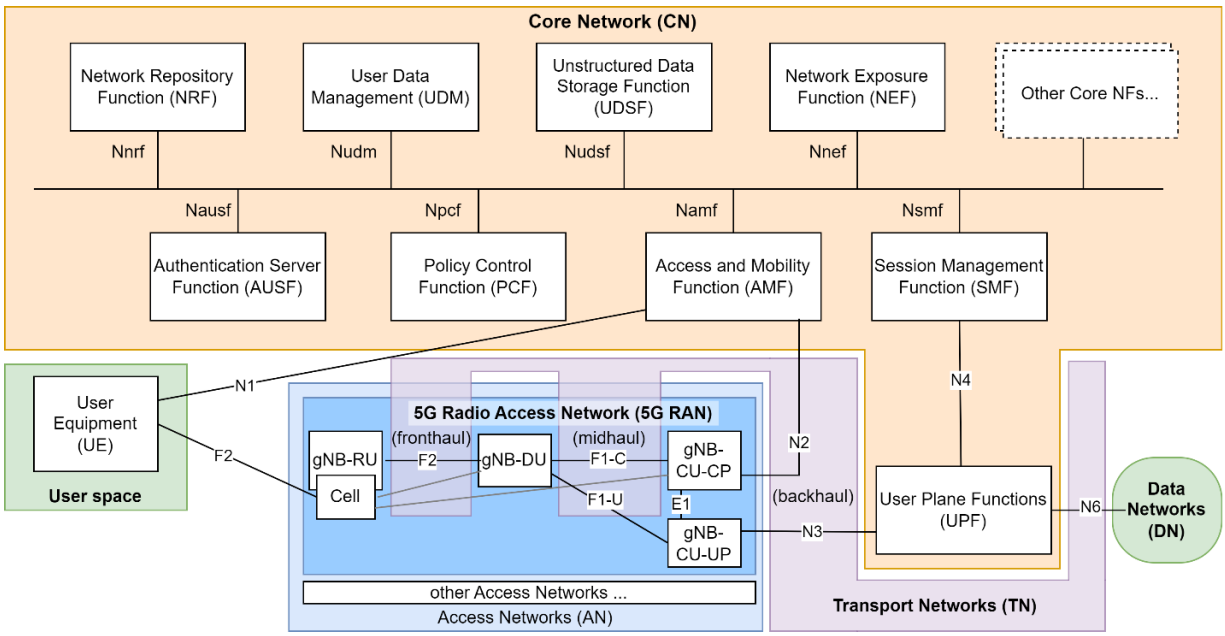


Figure 6. Functional architecture of the 5G network divided into the main domains: User space, Radio Access Network (RAN), Core Network (CN), Transport Networks (TN) and Data Networks (DN).

2.1.4. Organization of 5G Network Functions and Links in the RAN domain

RAN handles all radio-related functions, including user data transmission and reception over the air interface. There is a range of RAN types that are possible to be incorporated into the 5G network, out of which the 5G New Radio Access Network (NG-RAN, 5G RAN or simply RAN) deployed in the Standalone mode (SA)[1] is the most important in the context of this thesis. It is because NG-RAN offers the most advanced performance and management capabilities, including support for Network Slicing, which is not natively supported in the legacy RAN types. 5G RAN in the SA mode is built from so-called Next Generation Node B (gNB) base stations. The NFs building up the gNB installations include Centralized Unit (gNB-CU), Distributed Unit (gNB-DU) and Radio Unit (gNB-RU). Importantly, in the majority of realizations, the gNB-CU and gNB-DU NFs are provided as VNFs, making gNB-RU the only PNF among the entire spectrum of 5G NFs. Moreover, due to CUPS adoption, gNB-CU is further divided into two functional units: the Control Plane (gNB-CU-CP) and the User Plane (gNB-CU-UP). The functionalities and responsibilities of the particular gNB NFs are summarized in Table 2 below. The roles of the particular gNB NFs are important in the context of the discussions conducted at the later stages of this dissertation, including the PoC implementation of the thesis concepts.

Table 2. Main functionalities and responsibilities of the standardized 5G RAN Network Functions' types.

5G RAN NF	Functionalities and responsibilities
gNB-CU-CP	gNB-CU-CP is responsible for the Control-Plane functions, including: ○ **RRC (Radio Resource Control):** Manages the signaling between the UE and the network, including connection establishment, maintenance, and release. ○ **Mobility Management:** Coordinates handovers and mobility-related procedures to ensure seamless user experience when moving between Cells. ○ **QoS Management:** Ensures that appropriate QoS levels are maintained for different services and applications. ○ **Security Functions**: Handles encryption and integrity protection of control messages.
gNB-CU-UP	gNB-CU-UP is responsible for higher-layer processing of the User Plane traffic, including: ○ **PDCP (Packet Data Convergence Protocol):** Performs header compression, ciphering, and integrity protection for user data packets. ○ **Data Forwarding:** Routes user data to the appropriate UPF (User Plane Function) in the core network.
gNB-DU	gNB-DU handles lower-layer User-Plane processing and interacts with the radio interface: ○ **RLC (Radio Link Control):** Provides segmentation, reassembly, and error correction for data packets. ○ **MAC (Medium Access Control):** Manages scheduling, resource allocation, and hybrid automatic repeat request (HARQ) processes. ○ **PHY (Physical Layer):** Handles modulation, coding, and other physical layer functions. The PHY layer is responsible for the actual transmission and reception of data over the radio interface.
gNB-RU	gNB-RU interfaces directly with the UEs and handles the RF (radio frequency) processing, responsible for: ○ **RF Processing:** Manages transmission and reception of RF signals, including amplification and filtering. ○ **Beamforming:** Utilizes advanced antenna techniques to direct signals towards specific users, improving signal strength and reducing interference. ○ **Antenna Management:** Controls multiple antennas to enhance coverage, capacity, and overall network performance.

As depicted in Figure 6, there is a number of logical Links between the gNB NFs, which are also subjects to CUPS separation. Depending on the deployment arrangements, these Links can either be a part of the RAN domain or may be considered a part of the TN domain[26] if there is a significant physical separation between the placement of the gNB NFs (e.g. in the Cloud RAN architectures [76]). Due to this separation, the RAN Links are typically classified into three tiers: fronthaul (F2-C, F2-U), midhaul (F1-C, F1-U, E1) and backhaul (N2, N3).

2.1.5. Organization of 5G Network Functions and Links in the Core domain

The Core Network (CN, Core) is the heart of the 5G architecture, responsible for managing all aspects of the network's operation that enable its users to communicate. The NFs comprising the Core domain can be categorized into three groups:

- **User Plane** – composed of multiple User Plane Functions (UPFs) that are responsible for packet processing and traffic aggregation
- **Control Plane** – composed of NFs performing signaling that enables establishing and maintaining the user plane - signaling in this sense means the exchange of information that enables but does not provide the communication service itself. The control plane also includes gateways that separate the user and control planes while interacting with the UE
- **Data Plane** – allows for decoupling compute from storage. Thanks to the data layer, all NFs can be stateless, allowing for better scalability and resiliency

The functionalities and responsibilities of the most important subset of standardized Core NF types[27] are summarized in Table 3 below:

Table 3. Main functionalities and responsibilities of the standardized 5G Core Network Functions' types in the particular layers.

Layer	5G Core NF	Functionalities and responsibilities
User Plane	**User Plane Function (UPF)**	UPFs are responsible for forwarding user data packets and managing the data flow between the 5G Core Network and the external Data Network. UPFs handle data packets routing, forwarding and policy enforcement. They are also responsible for per-flow QoS handling.
Control Plane	**Network Repository Function (NRF)**	The NRF is crucial for the efficient operation of the 5G Core Network NFs, providing centralized registration, discovery, and status monitoring of NFs. It enables dynamic discovery of services offered by other NFs, ensures service availability, supports load balancing and enforces security policies over the interactions between the NFs.
	Access and Mobility Function (AMF)	The AMF manages user mobility and access, including registration, connection, and handover procedures. It handles signaling between UEs and Core Network.
	Session Management Function (SMF)	The SMF is responsible for session establishment, modification, and release. It manages IP address allocation, traffic routing policies, and interacts with the UPF for allocating the user plane resources.
	Policy Control Function (PCF)	The PCF provides policy rules to control plane functions, enforcing QoS and other network policies. It ensures that the network resources are allocated according to the defined policies and service agreements.
	Authentication Server Function (AUSF)	The AUSF handles authentication of UEs to ensure secure access to the network. It works with the UDM to authenticate users and devices.
	Network Exposure Function (NEF)	The NEF provides a secure and controlled way for third-party applications to access network services and capabilities. It acts as an intermediary between the 5G network and external applications, such as IP Multimedia Subsystem
Data Plane	**User Data Management (UDM)**	The UDM stores and manages subscriber data, including user profiles, authentication information, and subscription details. It interacts with other control plane functions to provide the necessary user data.
	Unstructured Data Storage Function (UDSF)	UDSF stores and manages unstructured data including information that is varied in format, such as logs, traces, and other non-tabular data. UDSF stores data generated by various network functions and services, providing efficient mechanisms for their storage and retrieval.

In terms of logical Links between Core NFs, they can be considered from two perspectives: service-based or Reference Point view. The upper part of Figure 6 presents the service-based view, schematically depicted as a communication bus, where each SBA-compatible NF can access the services offered by other SBA-compatible NFs through their Service-Based Interfaces (SBI). The SBIs of the particular NFs are named according to `Nxxx` pattern, where `xxx` is the abbreviated name of a given NF. For example, `Namf` is the AMF's SBI, through which services like `Namf_Communication` or `Namf_Location` are offered. Establishment of the concretized, point-to-point logical Links between SBA-compatible NFs is achieved via the Network Repository Function (NRF), which performs the role of a service registry, offering functionalities like NF registration (`Nnrf_NFManagement`) and discovery (`Nnrf_NFDiscovery`), allowing NFs to identify the services of other NFs they need to interact with.

[26] Due to the reasons provided in Section 1.6, The TN domain aspects are not in the main scope of this dissertation.
[27] See Clause 6 and 7 in [68] and Clause 5.2 in [72] for a full list of standardized 5G NFs.

While SBI enables the establishment of logical Links between the SBA-compatible NFs in any arrangement, in a typical scenario, interactions occur among particular sets of NFs, which can be presented in the form of the "traditional", point-to-point interactions through Reference Points as shown in Figure 7. In such a view, Links between the particular pairs of NFs are identified by the name of a logical Reference Point, e.g. N11 Link between AMF and SMF or N8 Link between UDM and AMF. Figure 7 schematically depicts the service-based interactions between services of AMF and SMF through the N11 Link.

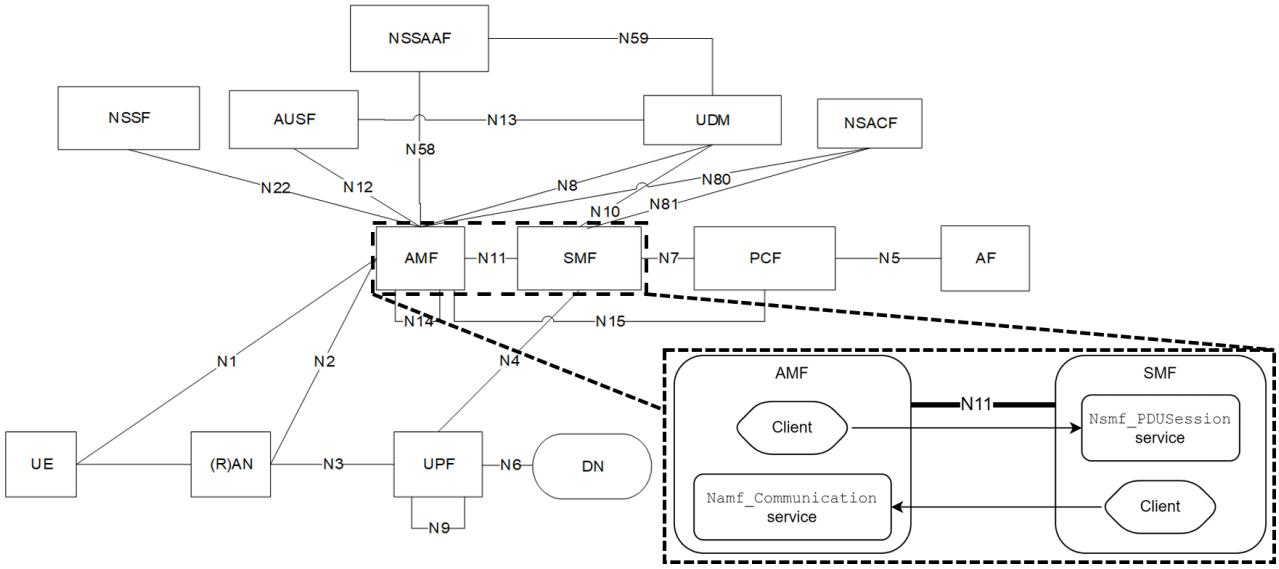


Figure 7. Relationships between 5G NFs through numbered Reference Points, including examples of Service-based interactions between the AMF and SMF through the N11 Reference Point.

Introduction of SBA in the 5G Core domain [28] constitutes an important enabler for Network Slicing management and orchestration. It is because NF instances comprising the Core deployment in a particular Slice can be flexibly added/removed/reconfigured to formulate a telecommunication network offering a particular range of functionalities, without interrupting the operation of the entire system. Moreover, SBA enables dynamic establishment of logical Links between the particular NFs in a plug-in-based manner, reducing the configuration burden[29] which is among the most challenging aspects to be embraced in the Network Slicing technology. Also, the CUPS separation plays a major role for slicing the NFs in both Core and RAN domains, because the majority of the Control Plane NFs can be reused among multiple Slices and only a limited number of Slice-dedicated NFs need to be maintained for the purpose of a dedicated, virtual telecommunication network that is perceived by its users as a logically separated one. Moreover, independent scaling of NFs in a particular plane can be performed.

2.1.6. Network Slicing as a feature of 5G

The architectural principles and NFs' organization as described in the previous Subsections serve as direct enablers for introducing the Network Slicing feature in 5G. Essentially, Network Slicing is about spawning, configuring and effective management of multiple sets of the mentioned NFs simultaneously on top of a shared cloud infrastructure, in order to satisfy the demanded characteristics of various Services, as schematically depicted in Figure 8.

[28] Currently, SBA is standardized in the Core Network domain only. However, there are attempts of applying it in the RAN domain as well [184].

[29] As a basic configuration of the SBA-compatible NFs, the NRF-repository identity and address need to be passed. Nevertheless, a range of other configuration items still need to be delivered to the NFs as a part of each Slice/Subnet provisioning – a comprehensive study of such configuration items in various Network Slicing building blocks is provided in Subsection 3.3.2.

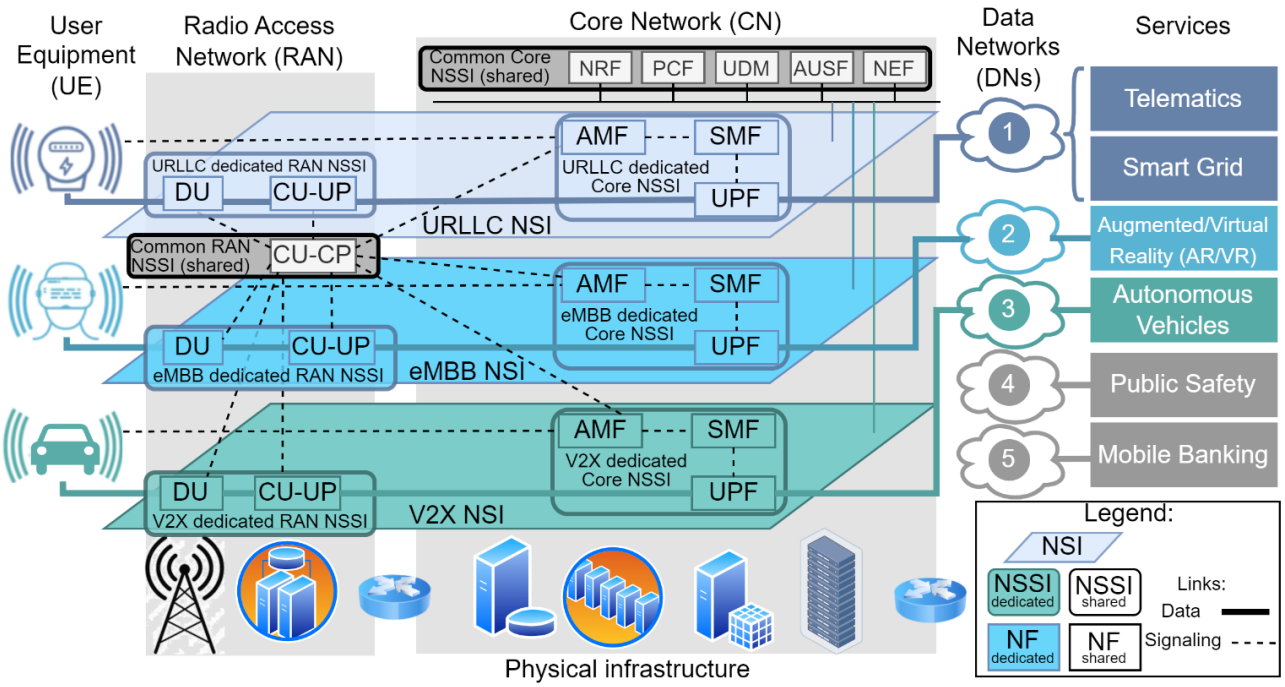


Figure 8. 5G network with several NSIs deployed, consisting of dedicated and shared NSSIs and NFs. Each NSI supports Services with different communication characteristics. Figure based on [77].

Each Slice is composed of Subnets which, in turn, contain NFs that may be **Slice-dedicated** (colored backgrounds) or **shared among multiple Slices** (white backgrounds). Whether an NF is dedicated to a Slice or shared depends on the performance requirements, isolation needs, and use case specifics. In the RAN domain, gNB-DU and gNB-CU-UP are often Slice-dedicated for proper isolation of the User-Plane traffic and its dedicated treatment according to the demanded Slice characteristics. Critical Core NFs like SMF, UPF, and sometimes PCF are typically Slice-dedicated to ensure optimal performance and isolation, whereas other functions like NRF, UDM, and AUSF are generally shared for central management.

While the telco networks themselves are complex structures, effective management of which is challenging, introducing the additional functionality of Network Slicing on top of the already existing features undoubtedly increases this complexity even more. To embrace it, appropriate models, systems and procedures are essential to be elaborated and uniformly adopted by all parties involved in the development and exploitation of the telco ecosystem. Such harmonized development and operation of telco networks is aimed to be achieved by the joint efforts of multiple industrial, academic and business stakeholders involved in the process of **standardization**.

2.2. Basic aspects of standards-aligned management of 5G networks

Discussing the organization of NSI/NSSI design process and its automation requires prior explanation of the basic aspects of telecommunication networks management in general, with a particular focus on their 5th generation. For this purpose, the most important aspects of the standardized management models, methods and systems utilized in 5G are presented in this Section. As discussed in Section 1.6, there is a number of SDOs involved in the standardization of various aspects of 5G and Network Slicing. However, these are the 3GPP and ETSI standards[21] that provide detailed specifications of the provisioning, management and orchestration aspects of 5G networks and, as such, are the most relevant in the context of this thesis. Thus, a comprehensive review of these standards is provided in the current and the following Section 2.3. The issues discussed in the current Section refer to 5G management in general, serving as an essential basis for introducing the Network Slicing-specific management aspects described in Section 2.3.

2.2.1. Conceptual models of elements comprising a 5G network

In order to properly outline the basics of standards-aligned Network Slicing management and orchestration, it is important to explain the 3GPP approach towards **modeling** the 5G NFs and Links that were introduced

in Subsections 2.1.4 and 2.1.5. The basic management concepts applicable to telecommunication networks in a range of generations are provided in [78]. This standard specifies, among others, the fundamental notion of a **Network Element (NE)**, defined as "*a discrete telecommunications entity which can be managed over a specific interface*". In turn, the concept of a **Network Function (NF)** in the context of 5G is introduced in [68] where it is defined as "*a 3GPP adopted or 3GPP defined processing function in a network, which has defined functional behavior and 3GPP defined interfaces"*. The same specification further notes that "*NF can be implemented either as a Network Element on a dedicated hardware, as a software instance running on a dedicated hardware, or as a virtualized function instantiated on an appropriate platform, e.g. on a cloud infrastructure*".

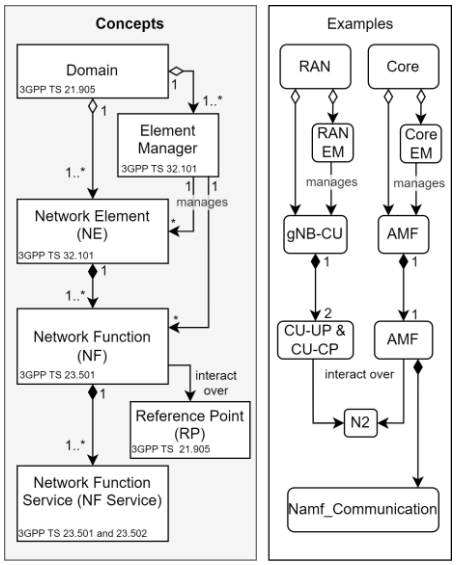


Figure 9. Basic telco networks management concepts standardized by 3GPP.

By contrasting the provided NE and NF definitions, it can be stated that while NE represents the concept of a managed telecommunication equipment as a whole (which can be hardware-based or virtualized), NF is the software application running on such equipment that provides its specific functionalities. With more granularity, the particular subsets of functionalities of a given NF (e.g. SMF) are modeled according to the SBA principles as Network Function **Services (NF Service)**[30] (e.g. `Nsmf_PDUSession`). In turn, the interactions between a given pair of NFs are performed over a **Reference Point (RP)**, which is defined in [79] [31] as a "*a conceptual point at the conjunction of two non-overlapping functional groups*". The concept of an Element Manager is discussed in Subsection 2.2.4. The relationship between the concepts of NE, NF, NF Service and RP, together with examples of representative instances of these concepts are schematically depicted in Figure 9[32].

The concepts introduced above are utilized in the fundamental 5G standards defined by 3GPP in [68], [72] for describing the detailed functionalities, responsibilities and interactions between all standardized 5G NFs. However, these standards do not define any models that could be utilized for NFs and Links management by an Orchestrator. Such models are defined in a set of corresponding, interrelated lower-level specifications, of which the most relevant information in the context of this dissertation is summarized in the following Subsection.

2.2.2. Management models of elements comprising a 5G network

A set of generic models that shall be utilized in 3GPP-compliant Orchestrators is defined in [80], which is referred to as the **Generic Network Resources Model** (**Generic NRM**). Among others, Generic NRM defines the following, basic management models that correspond to the conceptual models discussed in the previous Subsection:

- **`ManagedElement`** model (ME) is used to represent the management aspects of a Network Element (NE). Inside of an Orchestrator, instances of the `ManagedElement` model are used for communicating with a manager over management interfaces for the purpose of being monitored and controlled.
- **`ManagedFunction`** model (MF) is used to represent the management aspects of a particular Network Function (NF). Each `ManagedFunction` instance (MFI) within an ME instance communicates with a manager over management interfaces exposed via its containing ME instance for the purpose of being monitored and controlled.

[30] Detailed specification of the 5G NF Services is provided in Clause 7 of TS 23.501 [68] and Clause 5 of TS 23.502 [72].

[31] Relationship between Service-Based Interfaces (SBI) and Reference Points is explained in Annex A of TS 23.501 [68].

[32] When analyzing Figure 9, it is worth noting, that while typically NE hosts a single NF, there are cases in which multiple NFs are contained in a single NE. Such situation occurs, for example, in the case of "non-split" or "2-split" variants of gNB installations described in [32]. In the "non-split" variant, all 3 gNB NFs (DU, CU-UP and CU-CP) are provided within a single NE. In the "2-split" variant, the DU NF is provided by a dedicated NE, but the CU NFs (both CU-UP and CU-CP) are hosted in a single NE.

- **`Link`** model represents a communication link or Reference Point (RP) between two network entities - the represented communication link or RP may either be a physical or logical entity.
- **`EP_RP`** model, which stands for "End-Point-for-Reference-Point" represents an end point of a `Link` that is used across RP between two network entities (e.g. MEs or MFs).

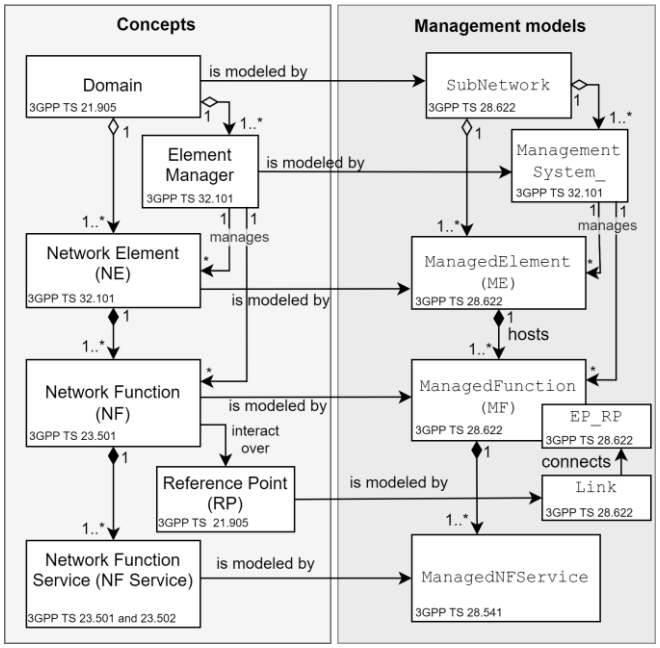


Figure 10. 3GPP Generic and 5G NRM management models corresponding to conceptual entities.

Furthermore, Generic NRM specifies the **`SubNetwork`** model, which represents the administrative domain in which the MEs are deployed and managed by means of one or more **`ManagementSystems`**. Since this dissertation focuses on the management aspects rather than on the particular functionalities of the NFs, from this point onwards, the entities building up a 5G network will most often be **referred to in their management nomenclature**, as MEs, MFs, etc. However, when a clear distinction between the conceptual and management perspectives is needed, the conceptual models' nomenclature will be utilized as well. The relationship between the enumerated management models and their correspondence to the conceptual models discussed in the previous Subsection are depicted in Figure 10.

All models defined in the Generic NRM are abstract - they serve as ancestors of generation-specific management models that derive from them and inherit sets of attributes that are common to every specialization of those models. In particular, the Generic NRM models are utilized as ancestors in a 5G standard that is particularly important in the context of this thesis, namely the **5G Network Resource Model** (**5G NRM**) provided in [32]. This specification is essential, because besides providing a broad spectrum of standardized management models for 5G `ManagedFunctions` and `EP_RPs` in both RAN and Core domains, it also defines the **essential Network Slicing-related management models** that will be discussed in details in the upcoming Subsection 2.3.3.

Among others, Clause 4 of 5G NRM specifies the models of MFs of the RAN domain that correspond to the NFs discussed in Subsection 2.1.4, namely **`GNBDUFunction`**, **`GNBCUCPFunction`**, **`GNBCUUPFunction`** – these models inherit from the abstract `ManagedFunction` model, containing sets of attributes that represent the management aspects of standardized gNB-DU, gNB-CU-CP and gNB-CU-UP NFs[33]. Similarly, Clause 5 of 5G NRM specifies the models of standardized 5G Core MFs that correspond to the NFs discussed in Subsection 2.1.5, for example `AMFFunction`, `SMFFunction` and so on.

Definitions of models in the 5G NRM are fundamental for constructing interoperable 5G systems, because they define the types, cardinalities and semantics of attributes that shall be exposed by the implementations of the particular 5G MFs. As such, the 5G NRM models[34] are commonly utilized by vendors to build their standards-compliant offerings of MFs implementations. Consequently, 5G NRM also serves as an important indicator for solutions architects who design the functionalities of the corresponding Orchestrators.

In particular, specifications of individual MFs models defined in 5G NRM are crucial for the organization of the NSI/NSSI design and allocation processes, because they **define sets of attributes characterizing a given MF that may potentially require manipulation of their values when designing and allocating an NSI/NSSI.** As an example, the value of the `pLMNInfoList` attribute[35], which is present in the majority of the standardized

[33] It is worth noting that the Cells being under the control of gNB-DU and gNB-CU-CP (see Figure 6 in Subsection 2.1.3) also have their management models deriving from the abstract `ManagedFunction` model: `NRCellDU` and `NRCellCU`, respectively.

[34] This includes the OpenAPI and YANG models' definitions that are attached in the TS 28.541 archive [32].

[35] More detailed explanation of this attribute's semantics, important to be taken into account in the design process, is provided in Subsection 5.3.3, in the context of the PoC implementation.

MFs' models, conveys information about the supported Slice types (SST[36]) by a given MFI and the possibility of NSIs/NSSIs allocation with such types by the particular Customers. Due to the mentioned facts, the management models defined in 5G NRM serve as one of the main references in the author's research towards proposing NSI/NSSI design process automation solutions elaborated in this thesis. In particular, a broad analysis of the standardized MFs attributes in terms of their relevancy for the NSI/NSSI design process, as well as their classification from several perspectives that are important for achieving design process automation is among the significant contributions of this thesis, discussed in Subsection 3.3.2. This contribution is especially important due to the fact that the design-time issues are left completely unspecified in the 3GPP standardization, which is extensively discussed in the upcoming Subsections.

It shall be noted that the management models in Generic and 5G NRMs are defined by 3GPP in a manner that is agnostic from the technical realization of a given ME. Thus, to provide a full management view of a given ME, these models need to be mapped to their **technical realization** as a PNF or VNF, as schematically depicted in Figure 11. In both realizations, MFs are implemented as software applications running in the runtime environment provided by their underlying PNF or VNF. In turn, the `ManagedNFServices` offered by a given MF correspond to the particular functionalities of those applications that are accessible by means of a standardized interface, e.g. a RESTful one.

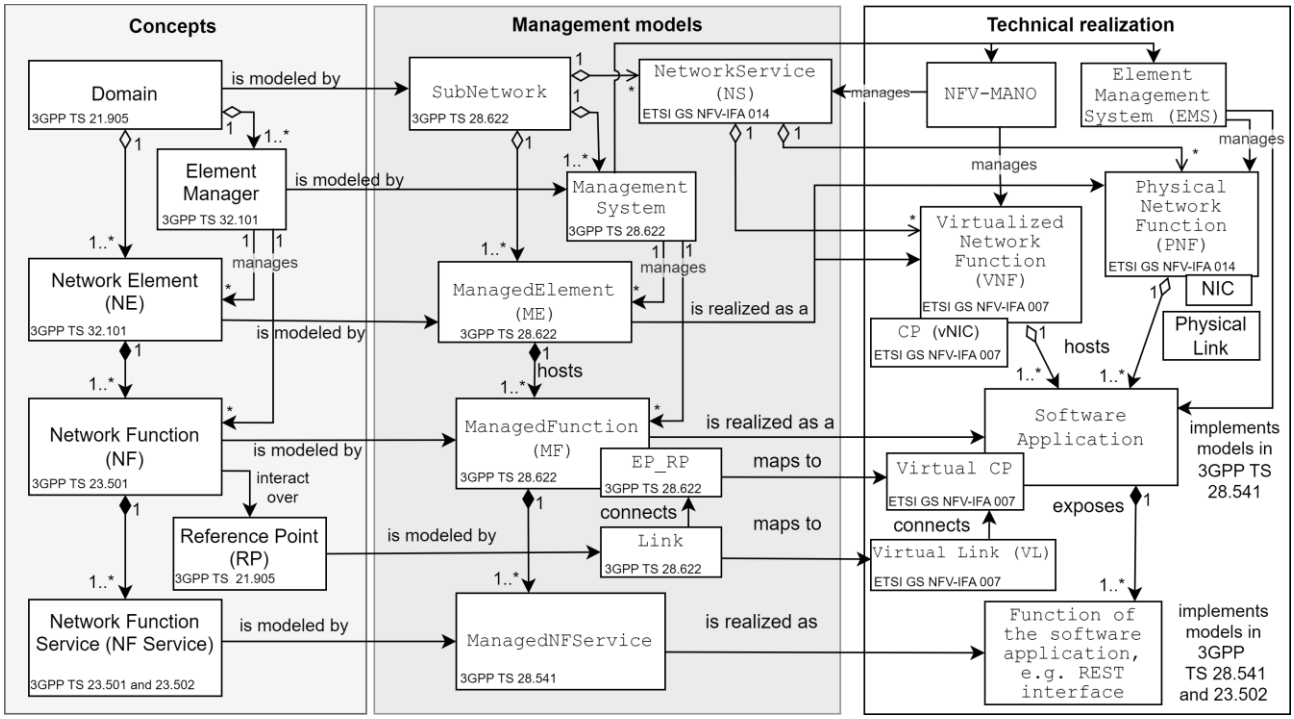


Figure 11. Correspondence between the concepts, generic management models and their concretizations within the 3GPP and ETSI NFV-MANO standards.

In the case of PNF-based MEs, the logical `EP_RPs` map to the physical Network Interface Cards (NIC) installed in the PNF's hardware, and the logical `Links` between `EP_RPs` are established on top of physical links connecting the NICs. Typically, management of PNF-based MEs not only requires manual interventions of the domain administrators to set up the hardware appliances, but is also performed according to vendor-specific models and methods that are tightly integrated with their corresponding, highly-specific `ManagementSystems`. For this reason, coverage of management aspects of the PNF-based MEs in the context of automation solutions described in this thesis is limited to their MF-layer only.

For MEs delivered in a form of VNFs, a comprehensive set of models and procedures of their handling is defined in a suite of **ETSI NFV-MANO standards**. These standards define both conceptual and management models of a VNF that is perceived as an encapsulated unit, as well as its internal components that provide a virtualized view of the underlying hardware elements. More specifically, [81] provides management models

[36] See the standardized Slice/Service Type (SST) values and their characteristics described in Subsection 2.1.1.

of a VNF perceived as a whole, as well as its **Connection Points (CP)** and **Virtual Links (VL)** that connect them[37]. In case of VLs that connect VNFs deployed in different DCs, the abstraction of a Multi-Site Connectivity Service (MSCS) defined in [82], [83] is utilized to model the underlying Data Center Interconnection (DCI). In turn, [84] specifies an important construct of a **Network Service (NS)** which constitutes a grouping of VNFs, VLs and nested NSs that from the management perspective are all treated collectively as a single unit. Due to the shift towards virtualization, the ETSI NFV-MANO standards are very relevant to the 3GPP standards in the 5G Releases and these standards are frequently referred to from the 3GPP specifications.

However, it is important to note that the NFV-MANO standardization focuses on the management of **VNFs that are perceived purely as runtime environments** in which MFs operate, without providing dedicated perception for handling their particular types. In other words, the ETSI NFV-MANO standards do not deliver dedicated models of VNFs for MFs that run inside of them (i.e. there is no definition of a SMF VNF, AMF VNF, gNB-DU VNF, etc.), but treat each VNF in a generic, **application-agnostic** manner, without "understanding" the semantics of the particular MF. Instead, NFV-MANO standards aim at defining the Lifecycle Management (LCM) of VNFs hosting MFs and exposing various features of the underlying virtualization platform[38] to be accessible by the MFs. As an illustrative example, while NFV-MANO systems monitor the faults and performance of a given VNF in terms of its CPU/memory/storage usage, these systems are not aware of any application-level performance measures, Key Performance Indicators (KPI) or faults, e.g. UE registration success rate in AMF, Protocol Data Unit (PDU) session establishment time in SMF, etc. It is because the high-level models of MF applications running inside of VNFs are not known to NFV-MANO[39].

As described above, there is a clear distinction between the 3GPP and ETSI NFV-MANO standardization scopes – NFV-MANO provides the means for managing virtualized execution environment for the operation of MEs, but the specifics of the MFs running inside of them are defined by 3GPP. However, despite the wide extent of covered aspects, there is an important gap at the junction of these two standardization scopes that relates to the design-time aspects of 5G networks management, which are discussed in the following Subsection.

2.2.3. Design-time models of elements comprising a 5G network

In the process of designing an NSI/NSSI the information about the building blocks that can be used to construct it is generally of two kinds:

1. characteristics and status of ME instances **already existing** in the 5G network that may be shared among Slices and hence can be considered for reuse;
2. specification of MEs that **do not exist yet** but are considered to be newly spawned.

The management models introduced in the previous Subsection reflect only the first perspective, and only to a partial extent. It is because the instances of those models represent the **runtime perspective** of MEs that are already existing and configured in a particular way.

But, in order to introduce changes in the managed 5G network, like the allocation of a new NSI/NSSI or deallocation of an existing one, it must be possible to express the planned changes in a systematic way. More specifically, it must be possible to (1) describe the full configuration of MEs that are planned to be newly created and interconnected with other MEs (existing or to be newly created), and to (2) express the decided changes to be applied in the configuration of the already existing ME instances (without actually introducing them). These two aspects constitute the **design-time perspective** of MEs and of the 5G network in which they operate.

Importantly, the design-time perspective of an individual ME pertains to each of its layers, including the VNF/PNF it is embodied in (including its CPs, VLs, etc.) and most importantly the MF applications installed within it. It also concerns the interconnected group of MEs combined into NSs, as well as their interoperation

[37] There are also other constructs like VNF Forwarding Graphs (VNFFG), etc. which were considered in the early perception of Network Slicing. However, their usage is not mentioned in the contemporary Network Slicing standards anymore. Thus, their elaboration is omitted.

[38] These are the features of the cloud management software like OpenStack [185], Kubernetes [186] and of the underlying hardware, e.g. hardware acceleration modules, Single Root Input-Output Virtualization (SRIOV) [155], etc.

[39] CM/PM/FM of the MFs is the responsibility of Element Management Systems (EMS) that are discussed in the upcoming Subsection 2.2.4.

with other MEs already operating in a given 5G network. Standardization covers the design-time perspective only partially, because **design-time models are only available for the NFV layer specified by ETSI, but are not provided for the MF management layer specified by 3GPP**. The most important aspects of the existing NFV design-time models, and the consequences of lack of 3GPP design-time models are discussed in the following paragraphs and in the upcoming Subsection 2.3.4, where the lack of standardized Slice/Subnet-level design-time models is discussed.

All NFV management models introduced in the previous Subsection (i.e. VNF, NS, CP, VL, etc.) have their corresponding design-time models that enable both to plan the provisioning of their new instances and to describe the changes to be introduced in the existing instances. The first perspective is provided by so-called **Descriptors** that are corresponding to each entity mentioned earlier. More specifically, [81] specifies a generic model of a **VNF Descriptor (VNFD)** that allows for specifying the structure, resources requirements (CPU, memory, storage, networking interfaces), deployment policies and LCM rules for the described VNF. PNFs that need to be integrated into the NFV-MANO framework can have their **PNF Descriptor (PNFD)**[40] too, which is specified in [84]. In the case of multiple tightly-related MFs, a **Network Service Descriptor (NSD)** specified in [84] is utilized for defining how their underlying VNFs/PNFs and nested NSs shall be interconnected and managed as a group. The VNFDs and NSDs are utilized as an essential input for the NFV-MANO systems[41] to perform VNFs and NSs instances deployment, their initial configuration (Day 1) and operational management (Day 2). A schematic overview of the structure of a VNFD and an NSD is depicted in Figure 12 and their most important parts in the context of this thesis are briefly described below.

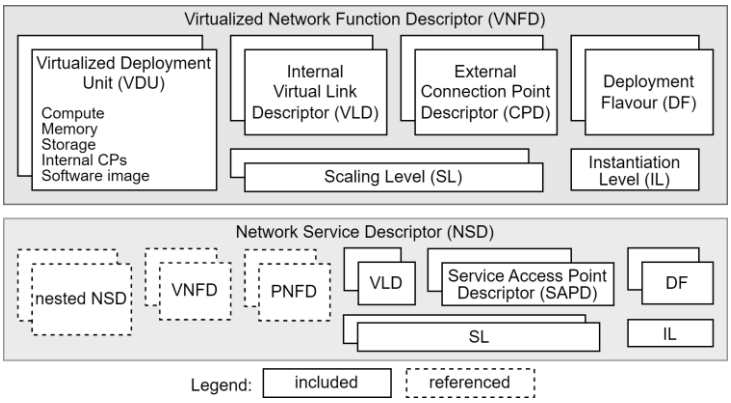


Figure 12. High-level overview of the ETSI NFV-MANO VNF Descriptor (VNFD) and Network Service Descriptor (NSD) structure.

Internally, VNFD contains so-called Virtualization Deployment Units (VDU) which specify the compute, storage, and networking resources needed for internal units of a VNF, referred to as its VNF Components (VNFC). Each VNFC instance, created based on the VDU specification, maps to an atomic deployable unit which can be a single VM in the case of hypervisor-based virtualization, or one or a set of containers in the case of OS virtualization. VNFD also contains Virtual Link Descriptors (VLD) and Connection Point Descriptors (CPD) that describe the internal connectivity between VNFCs and external connectivity with other VNFs/PNFs and NSs. In turn, NSD includes references to the VNFDs/PNFDs constituting the described group, VLDs for describing the topology of interconnections between them as well as exposition of selected VNFs' CPs to the outside of the NS, specified by means of Service Access Point Descriptors (SAPD).

Instances of the VNFD/NSD model represent the design-time specification of a given VNF/NS. Particular VNFD/NSD instances are typically delivered in a machine-readable format of the TOSCA modeling language[42] utilizing constructs standardized by ETSI in [85]. VNFD together with the MF software image and other files necessary for its operation (scripts, datasets, configs, etc.) are recommended to be bundled into a **VNF Package** according to the standardized structure of a Cloud Service Archive (CSAR) dictated by [86]. Typically, VNF Packages and the lowest-level NSDs (not referencing any nested NSDs) are crafted by the Vendors of the particular MFs, taking into account the operational requirements of the Provider and the particularities of the utilized Orchestrator. Such adjustments may result in the inclusion of specific, non-standardized **configurable extensions** in the structure of Descriptors, with the aim of integrating a given VNF/NS in the Provider's network environment and its proper functioning. In turn, higher-level NSDs that encompass broader groups of MFs, entire sub-networks or even the entire 5G network, are typically crafted by the

[40] Due to the author's focus on the management of virtualized MFs (VNFs), the PNF related aspects are not elaborated for brevity reasons.

[41] Systems utilized for the management of individual 5G MEs and Network Slices are discussed in the upcoming Subsections 2.2.4 and 2.3.5.

[42] TOSCA stands for Topology and Orchestration Specification for Cloud Applications [163]. TOSCA is a text-based language that allows for modeling the deployment and management of cloud-based applications and services in a vendor-agnostic, portable, and interoperable way.

Operators to fit their management objectives in the administered 5G network – this aspect is particularly important in the context of this thesis, as the generation of appropriate NSDs is among multiple design activities that need to be automated for achieving *declarative style* of NSIs/NSSIs allocation, as discussed in detail in the upcoming Subsections 3.3.5 and 3.3.6.

Important constructs of the VNFD and NSD models, which are relevant in the context of preserving Network Slicing QoS guarantees and for elaborating solutions automating the NSI/NSSI design process, are the **Deployment Flavors** (DF), **Scaling Levels** (SL) and **Instantiation Levels** (IL). DF in the VNFD is a specification for instantiating the VNF with different arrangement of the underlying resources, allowing the Operator to deploy the same VNF in different configurations[43] to meet various functional, performance, capacity, security or cost requirements. In the case of an NSD, DF dictates the deployment configuration for all internally referenced and contained elements. In turn, SL of a given DF specifies how many instances of the resources configured in a given DF shall be utilized. Thus, while DF defines the range of resources the VNF/NS can use, an SL specifies resources allocation within that range. Finally, IL is a designated SL that is applied at the moment of instantiating a given VNF/NS, determining its initial performance profile and resources footprint.

Definition of DFs, SLs, ILs and any vendor or operator-specific configurable extensions inside of the VNFD/NSD enables flexibility and adaptability of a given VNF/NS for operation in various network scenarios, such as high-performance, resource-constrained or energy-efficient configurations. These constructs of the underlying VNF/NS allow a given MF or a group of MFs to operate not only with particular non-functional characteristics, like performance, capacity, security or cost[44], but also with different sets of offered functionalities (e.g. a given DF may include a supporting Deep Packet Inspection (DPI) VNFC, but other DFs may not include it). As such, these configurable items of Descriptors in the NFV layer offer important features in the context of Network Slicing realization, which is pointed out among others in [87].

However, an important remark in the context of NSI/NSSI design process automation needs to be stated at this moment, which relates to the Descriptors' **concretization level**[45] in general. Due to the fact that Descriptors may offer multiple DFs/SLs and are frequently equipped with various vendor-specific or Operator-requested configurable extensions, a given Descriptor **alone** typically does not constitute a fully concretized, self-contained input that can be immediately utilized for creating an instance of the described entity. It is because an instance can only be created after delivering the fully concretized values for all of such configurable knobs, which the author refers to as **instance-specific configuration**[46]. Besides the thoughtful choice of a DF and IL, examples of such instance-specific configurations may include, but are not limited to: providing the name of the instance to be created, selection of the IP addressing scheme, selection of the deployment location in so-called Point of Presence (PoP), passing the IP/DNS identification of the EMS managing the given ME, passing identification of existing MEs with which to integrate, etc.

On the one hand, NFV-MANO standards deliver well-defined means for providing the instance-specific configuration for creating and instantiating a VNF/NS instance with a particular configuration, which is achieved by means of the `CreateVnfRequest` and `InstantiateVnfRequest` [88], as well as `CreateNsRequest` and `InstantiateNsRequest` [89] data structures. These data structures require passing the mandatory configurations, like the choice of DF and IL, and allow for passing values for the aforementioned extensions[47]. However, on the other hand, **conscious decisions about the particular choices of instance-specific configuration values must be made by someone, based on some criteria.** In the traditional, *imperative style provisioning* approach, such decisions are made by human Operators based on their expert knowledge, taking into account the conditions in the administered network and the operational requirements to be met by the instances to be created. The need for making such decisions in a thoughtful manner based on multiple

[43] As an example, depending on the definition of DFs in the VNFD, a given VNF may be deployed with an internal VNFC that provides a dedicated database service or without it, which will entail maintaining in-memory databases in the other VNFCs.

[44] However, the DF IL specification only dictates the resources configuration, without specifying what level of QoS is achievable – this gap is addressed by the original concept of *Slicing Entities Capabilities* introduced in Subsection 3.1.2.

[45] Level of concretization is an important characteristic in the taxonomy of *Slicing Entities* discussed in Subsection 3.1.1.

[46] Therefore, in the taxonomy of *Slicing Entities*, presented in the upcoming Subsection 3.1.1, the author classifies Descriptors as *not fully concretized Design-time Slicing Entities*, because they require potential contextualization and concretization before their operational utilization becomes possible.

[47] According to the taxonomy of *Slicing Entities* elaborated in Subsection 3.1.1, a Descriptor supplemented with the instance-specific configurations constitutes a *fully concretized* ***Instance Descriptor***, based on which an Orchestrator is able to perform instance commissioning.

sources of information is among the most important challenges to be tackled for achieving automated design process that supports declarative allocation of NSIs/NSSIs. Concepts enabling such automation are provided in the upcoming Chapters of this thesis.

Besides specification of Descriptors and models allowing for their concretization, ETSI also provides models that allow for conveying the second aspect of the design-time perspective, namely **expressing the changes to be applied to existing VNF or NS instances**. For example, [90] defines an operation of changing the DF of a VNF, which is performed based on the information delivered in the `ChangeVnfFlavourRequest` model. It must be admitted that the coverage of the ETSI standards for the NFV layer of MEs is broad, detailed and comprehensive[48], contributing towards the easier construction of NFV layer management solutions that are interoperable in terms of both runtime and design-time aspects.

However, in contrast to the comprehensiveness of the ETSI specifications in the NFV layer, the design-time perspective is completely abandoned by 3GPP for the MF application layer. Neither the 5G NRM nor any other 3GPP specification provides models for describing MFs and logical Links **before they are provisioned and allocated**, nor for **describing the decided configuration changes to existing MFs and Links that are to be applied**. Thus, how to convey the design-time perspective of the particular MFs or Links remains a challenge to be tackled individually by the implementing parties. The same gap in terms of lack of coverage for the design-time perspective is unfortunately present in the Slice/Subnet modeling area, consequences of which for the general Network Slicing management and automation of the Slice/Subnet design process are further discussed in the upcoming Subsection 2.3.4.

Importantly, as described in the following Subsection, there are separate systems responsible for the management of the MF and NFV layers. This means that configuring a single ME consists of commissioning procedures that need to be coordinated among these systems. In particular, each system needs to be fed with dedicated input that is related to the ME layer it manages. This fact incurs further complications when automation of NSI/NSSI design and allocation is tackled.

2.2.4. Systems utilized for 5G networks management and orchestration

Management of MEs in a 5G network is performed by a number of systems that are responsible for supervising their various aspects. 3GPP in [78] defines the concept of an **Element Manager** which *"provides a package of end-user functions for the management of closely related types of Network Elements. Element Manager's functionalities can be divided into: management of Network Elements on an individual basis and SubNetwork management that may include relations between the Network Elements, enabling additional functions on the SubNetwork level, e.g. network topology presentation, alarm correlation, service impact analysis, etc."*.

Typically, the functionalities assumed by this conceptual management entity are implemented in so-called **Element Management Systems (EMS)** that are developed by the particular vendors to enable management of MEs in the given domains, e.g. a RAN-dedicated EMS or a Core-dedicated EMS. EMS is responsible for the entire spectrum of FCAPS aspects of MEs as well as the MFs hosted by them. This includes tracking their current configuration (Configuration Management, CM), monitoring their faults (Faults Management, FM) and performance metrics (Performance Management, PM) including calculation of KPIs delivered by the particular MFs[49]. Among its key functionalities, EMS serves as the **main gateway for interacting with the MF software applications** running inside of MEs[50] - hence, at the lowest level, applying the design decisions in the existing and to-be-newly-created MFIs is performed via the EMS.

A typical EMS maintains an online repository of `ManagedElement` objects, their contained `ManagedFunction` and `ManagedNFService` objects. Such objects are instances of vendor's defined models that derive from the

[48] A collective specification of the standardized NFV-MANO models from all of the referenced specifications and beyond, including the conceptual, management and design-time models, is provided by ETSI GS NFV-IFA 015 [187].

[49] There is a standardized set of 5G KPIs defined in [188]. The KPI values constitute an important source of information for assessing the SLS Requirements' fulfilment by the particular Network Slice/Subnet building blocks and, as such, serves as an important indicator utilized in the NSI/NSSI design process, which is discussed in the upcoming Subsections 3.3.3 and 4.4.2.

[50] In the legacy networks, such interactions have been performed by means of the Simple Network Management Protocol (SNMP) [189]. This technology, however, is being replaced by more modern protocols like NETCONF [190] or RESTCONF [191], which utilize YANG models [91] for representing the management aspects of a given MF. YANG models are typically defined by MF Vendors based on 3GPP models discussed in Subsection 2.2.2.

standardized ones discussed in the previous Sections, e.g. `VendorA-SMF` model deriving from the standardized `SMFFunction` model. A given EMS is responsible for the management of MEs/MFs in the `SubNetwork` inside of which it is deployed, serving as the implementation of the conceptual `ManagementSystem` (see Figure 10 in Subsection 2.2.2). To perform its duties, **EMS needs to be equipped with precise models of MFs to interact with**, which are typically expressed by means of the YANG [91] modeling language.

While the MF software applications in 5G are managed by EMS, the kind of system utilized for the management of the MEs inside of which they are running depends on their technical realization. In case of PNF-based MEs, setting up the necessary equipment (e.g. antennas, bare metal machines, cables, etc.) and the basic underlying software (OS, libraries) is performed manually and with a potential support of out-of-bound management tools provided by the ME vendors[51] - such management aspects are out of scope of this dissertation. But, due to the shift towards virtualization in 5G, vast majority of MEs are nowadays delivered in the form of VNFs that require a dedicated cloud runtime environment and dedicated management systems. Specification of such systems is comprehensively covered by ETSI in the suite of NFV-MANO standards. A high-level overview of the systems comprising the NFV-MANO architecture is depicted in Figure 13 and described in the following paragraphs.

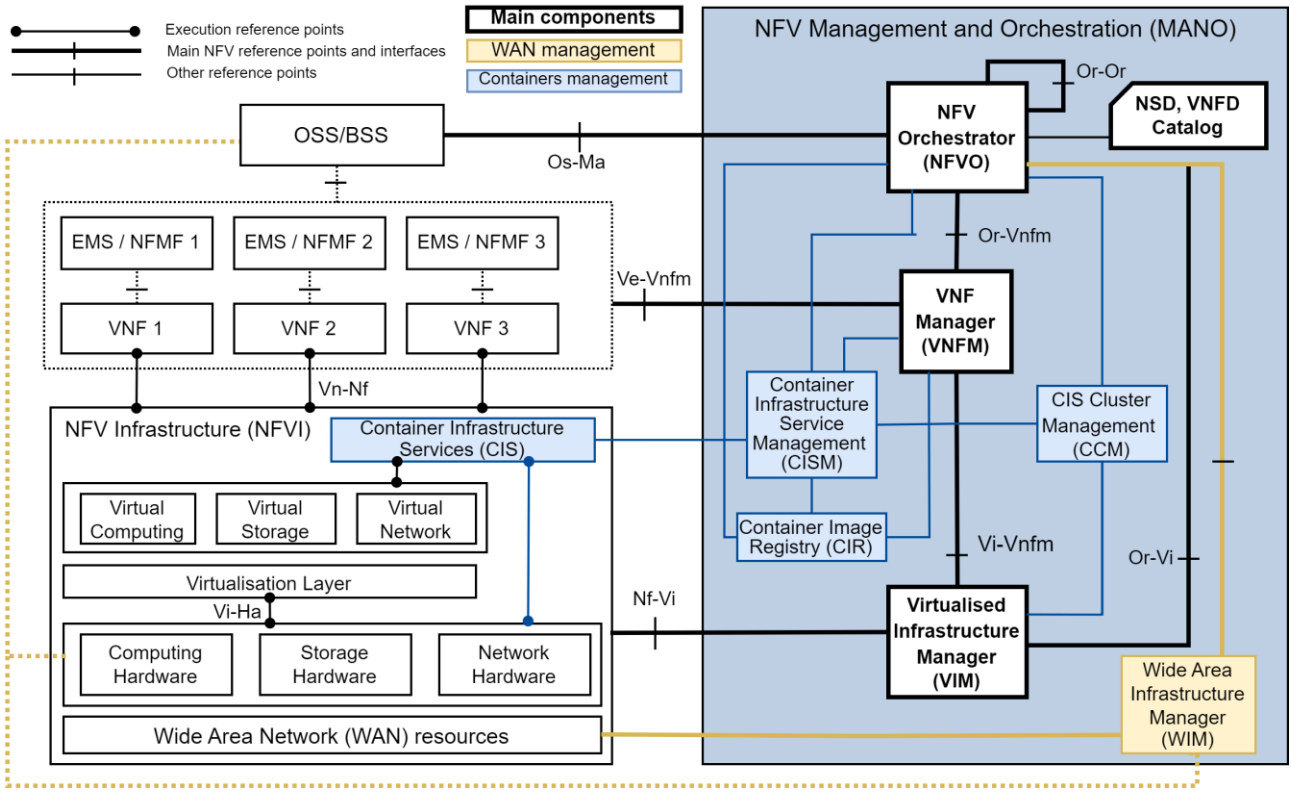


Figure 13. ETSI Network Function Virtualization Management and Orchestration Framework (NFV-MANO).

The primary functional blocks of the NFV MANO architecture are the NFV Orchestrator (NFVO), the VNF Manager (VNFM), Virtualized Infrastructure Manager (VIM) and the Wide Area Infrastructure Manager (WIM). From Release 4 of the NFV-MANO standards, these blocks have been supplemented with Container Infrastructure Service Management (CISM), Container Infrastructure Services (CIS), CIS Cluster Management (CCM) and Container Image Registry (CIR). The functions and responsibilities of these blocks are as follows:

- **VIM** is responsible for managing the physical and virtual resources of the NFV Infrastructure (NFVI). This includes compute, storage, and network resources within a data center and the cloud platform running on top of it, e.g. OpenStack or Kubernetes.
- **VNFM** is responsible for the LCM of individual VNFs, including their instantiation, monitoring, scaling, updating, and termination – as such, VNFM functionalities resemble the tasks that in the case of PNF-

[51] As an example, iDRAC [192] is an out-of-bound management tool in Dell servers, allowing remote management of physical machines – powering them up/down, accessing their outputs and inputs, sensors monitoring (e.g. fan speed, power voltages), etc. Such functionalities may also be integrated with EMS.

based MEs would need to be performed on the hardware manually by a human Operator. VNFM ensures that VNFs are deployed within the parameters specified in their VNFDs.

- **NFVO** is responsible for the overall orchestration and management of VNFs and their interrelated groupings in the form of NSs. NFVO is responsible for the LCM of NSs, including coordinated instantiation, scaling and termination of its comprising VNFs and VLs that interconnect them. NFVO manages the inventory of NSs, VNFs, and a catalog of their descriptors (NSDs, VNFDs).
- **WIM** manages WAN resources to provide an abstraction of Multi-Site Connectivity Service (MSCS), on top of which VL across different DCs can be established. As such, WIM ensures that VNFs and PNFs distributed across multiple PoPs can function together, within a single NS.
- **CISM, CIS, CCM** and **CIR** are extending the previous blocks with the capabilities of managing the container-based VNFs (so-called CNFs[23]).

The particular functionalities of the blocks enumerated above are exposed by means of their NBIs that are precisely defined in their corresponding specifications. NBIs of the lower-level systems are utilized in the Southbound Interfaces of the higher-level systems at the reference points. For example, as depicted in Figure 13, VNFM communicates with VIM over a standardized reference point called "Vnfm-Vi", specification of which is defined in [92]. Similarly, NFVO communicates with VNFM over the "Or-Vnfm" reference point, details of which are specified in [90] and [88].

Again, it needs to be underlined that the enumerated NFV-MANO functional blocks by themselves are not aware of the specifics of the MF software applications running inside of the VNFs or NSs containing them – these are the EMSs equipped with MF-specific models that assure such awareness, making EMSs essential blocks of the NFV-MANO architecture as depicted in Figure 13. However, EMS implementations utilized in the 5G networks are frequently reused from the previous generations, which supported PNF-based MEs only and, as such, do not natively support interactions with the NFV-MANO systems. But such interactions are needed for the management of the NFV layer underlying the MFs, which in the NFV-MANO architecture is possible through the "VeEn-Vnfm" reference point [93], [94].

As a common practice, another functional block called **Network Function Management Function (NFMF)** is introduced to fill this gap. To reuse the already implemented functionalities of the legacy EMS systems and to maintain their backward-compatibility, NFMF is typically implemented as a "wrapper"[52] that interacts with the VNFM block for VNF provisioning and re-exposes the EMS's functionalities of MFs management[53]. Such organization of the functional blocks enables the NFMF, among others, to trigger LCM operations based on some MF application-level criteria, e.g. changing the VNF's SL based on MF's performance metrics or KPIs, healing the VNF in case of MF malfunctioning, etc. While these are examples of operations that are important in the context of Network Slicing, their triggering requires either manual interventions of the NFMF's human Operator, or providing some kind of design-time specification that would dictate the logic for executing such operations. However, as already discussed in the previous Subsection, design-time models of MFs that would allow for expressing such specifications are currently not provided by any standards.

In the context of the design process automation tackled in this thesis, it needs to be pointed that the **NFV-MANO orchestration blocks work according to the *imperative provisioning style* principles**. More specifically, in order to commission the lifecycle of a given entity, a fully concretized design-time description in the form of a Descriptor (NSD/VNFD) supplemented with instance-specific configurations needs to be delivered, which is performed in a series of NBI invocations, as detailed in the following Subsection 2.2.5. **The responsibility for providing such a fully concretized design-time description is ceded to a higher-level entity – a human Operator or a higher-level management system**[54], like the Operation/Business Support System (OSS/BSS) depicted in Figure 13.

[52] Dedicated, standalone NFMF can be implemented as well.

[53] Additionally, in order to adhere to the principles of Service-Based Management Architecture (SBMA) standardized by 3GPP in [71], NFMF typically exposes its functionalities in a service-based manner by implementing the "NF provisioning service" defined in [31].

[54] In Subsection 2.3.5, where the Network Slicing management systems are introduced, it is justified why the Network Slice Subnet Management Function (NSSMF) is responsible for the delivery of the fully concretized design-time descriptions to the underlying NFV-MANO systems.

2.2.5. Basic management procedures related to individual or groups of elements comprising a 5G network

While the extent of functionalities offered by the management systems discussed in the previous Subsection can be very broad, some of the standardized management procedures that are relevant in the context of this dissertation need to be described. In particular, when there is a need for spawning new MFs with a Slice-dedicated configuration and/or modifying the existing, sharable MFs to prepare them for handling the newly allocated NSI/NSSI, a range of management operations need to be performed to ensure that the MEs containing such MFs are deployed, configured, and integrated into the existing network environment. Such operations are generally referred to as **Day -2, -1, 0, 1** and **2 management procedures**, which refer to the ME's release, pre-provisioning, provisioning, initial configuration and runtime exploitation, respectively. Importantly, the management activities pertain to each layer of a given ME: down from the VNF or PNF it is delivered as, up to the MF application layer that is running inside of it.

Traditionally, manual or assisted human Operator's involvement has been indispensable at the majority of the enumerated management processes. Sequence Diagram 1 illustrating the involved systems, actors, management steps they perform and information they need in order to make a single MF or a group of MFs (organized into an NS) operational, with the involved procedures described below the diagram. Analysis of these procedures allows for **highlighting the parts that are important to be taken into account in the NSI/NSSI design process and attempts of its automation**. In particular, the information items required at each step are pointed and the potential obstacles for automated execution of the relevant operations are identified. Importantly, the management activities pertain to each layer of a given ME: down from the VNF or PNF it is delivered as, up to the MF application layer that is running inside of it.

Day (-2) activities – Preparation of MF software, its Descriptors and models (not to be automated):

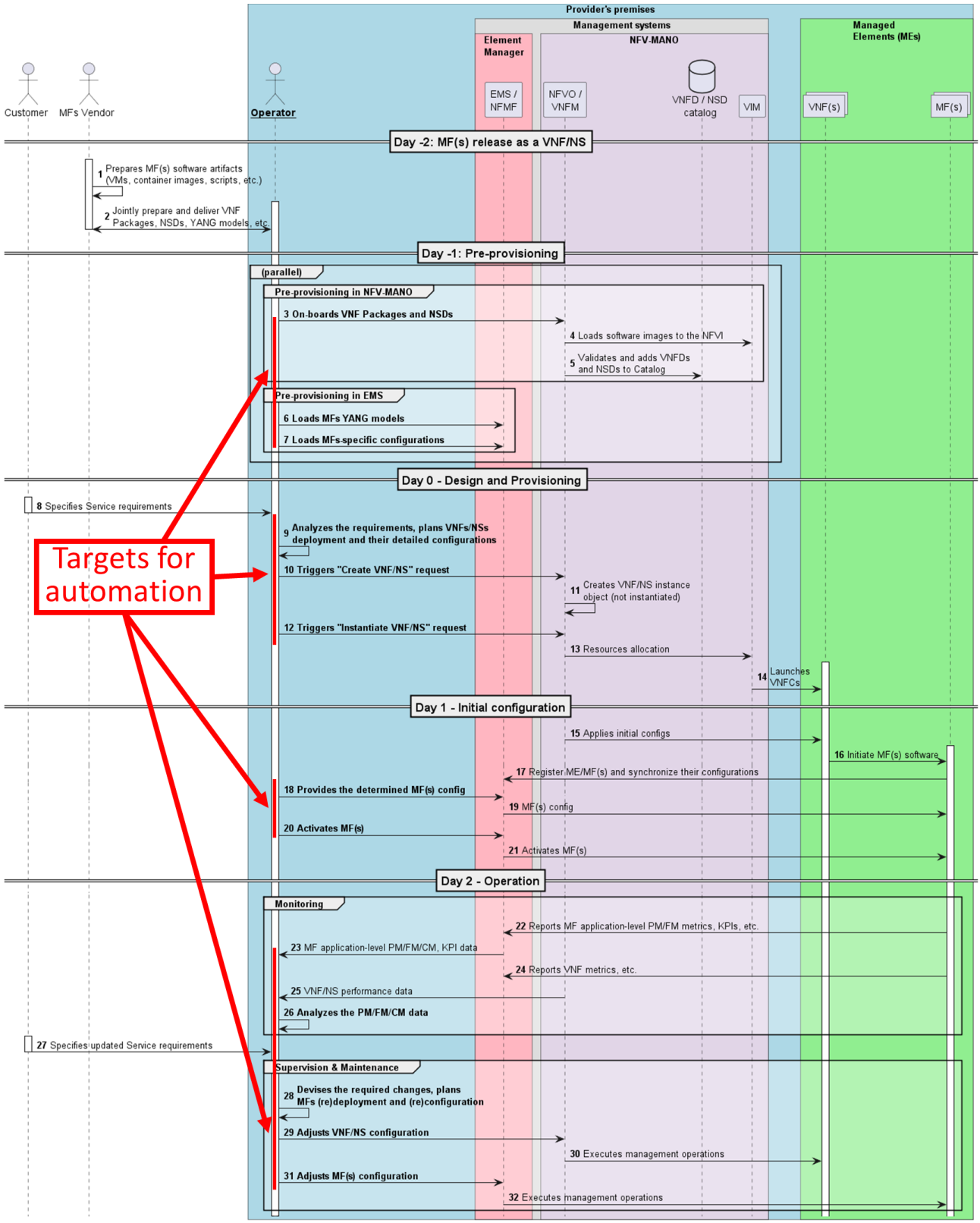
- Step 1: Vendor releases the MF software images (e.g. delivered as VM or Docker[55] images) and corresponding files (scripts, data files etc.):
  - Vendor, in cooperation with Operator, crafts instances of design-time models (typically files) required for MF's operation in the Provider's management systems: VNFD(s) describing the resources and runtime environment of the MF(s), and potentially NSD describing the arrangement and topology of interactions between a suite of multiple MFs.
  - In VNFDs/NSD, various **configuration items that require concretization during the design process may be included**, e.g.: IP addressing scheme, EMS/NFMF IP address or of other systems, etc.
  - YANG model(s) describing the **attributes of the MF application**(s), which may contain both standardized and vendor-specific extensions that **potentially require configuration during the design process** (e.g. application of a specific operational profile, QoS settings, etc.).
  - Vendor prepares VNF Package(s) and NSD CSAR[56] archives that contain the aforementioned files.
- Step 2: Vendor delivers the VNF Packages, NSD and YANG models to the Operator.

Day (-1) activities – Pre-provisioning (potentially to be automated):

- Steps 3-5: **Operator on-boards the VNF Packages and NSD** into NFV-MANO:
  - This operation validates and unpacks the VNF Packages, delivering their elements to the particular systems' repositories, e.g. VNFDs and NSD are placed in the NFVO's catalog, MF software images are transferred to the storage of the NFVI and the corresponding files are put in the VIM's repository, etc.
- Steps 6-7: **Operator supplies EMS with the definition of YANG models** corresponding to the management model of a given MF and any **MF-specific extensions**:
  - YANG models are utilized for the communication with the particular MEs and their contained MFs, for maintaining their operational view in the EMS repositories, for executing configuration changes operations as well as for fetching PM/FM data to the Orchestrator.

[55] Docker (https://www.docker.com/) is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software as containers.
[56] According to ETSI GS NFV-SOL 007 specification [90], Each DF of an NSD is represented by a separated file. Thus, an NSD with all its DFs shall be packaged as a CSAR archive that constitutes the logical NSD representation as an encapsulated, single file.

Sequence Diagram 1. Management steps needed to make a single MF, or a group of MFs (grouped into an NS), operational. These steps are divided into procedures executed in the particular phases: MFs' preparations by Vendor (Day -2), MFs' pre-provisioning in the Provider's infrastructure (Day -1), MFs' provisioning (Day 0), MFs' initial configuration (Day 1) and MFs' operation (Day 2).



Day 0 activities – Design and provisioning (to be automated):

- Steps 8-9: Based on the **Customer's requirements**, **Operator plans the VNF/NS deployment and configuration that will satisfy these requirements**.
- Steps 10-11: **Operator triggers creation of the VNF/NS stub objects** by invoking the "Create VNF/NS" operation exposed in the VNFM [88] / NFVO [89] NBIs:

- **Operator needs to decide and pass values for configuration items** required by the `CreateVnfRequest`/`CreateNsRequest` data types, including the IDs of the onboarded VNFD/NSD, the names of the instances to be created and configuration values that contextualize the attributes contained in the VNFD/NSD.
- "Create VNF/NS" does not instantiate the VNF/NS instance yet - this operation ensures that there are resources needed for the instantiation of a VNF/NS based on the configuration in the pointed VNFD/NSD. As a result, NFVO creates a new NS and/or VNFs objects inside of its operational inventory.

- Steps 12-16: **Operator triggers instantiation of the VNF/NS** by invoking the "Instantiate VNF/NS" operation exposed in the VNFM/NFVO NBIs:
  - **Operator needs to decide and pass values for multiple configuration items** required by the `InstantiateVnfRequest`/`InstantiateNsRequest`, including the choice of DF, IL, PoP and multiple other standardized settings (specified in [88], [89]) and **vendor-specific extensions**.
  - During the instantiation procedure, the resources required for VNF(s) operation are allocated, the VMs/containers are spun up and the contained MF(s) software image is launched. After successful instantiation, the VNF(s) and its contained MF(s) are running.
- Step 17: ME(s) and MF(s) objects register in their corresponding EMS/NFMF and their current configuration is synchronized to EMS/NFMF.

Day 1 activities – Initial configuration (to be automated):

- Steps 18-19: **Operator decides on** and **triggers MF's initial configuration** to prepare it for operation with other MFs and allow it **to meet the anticipated characteristics, e.g. fulfillment of specific QoS requirements**.
- Steps 20-21: **Operator activates the MF(s)**.

Day 2 activities – Operation and supervision (to be automated):

- Steps 22-25: VNF's and MF(s)' operational PM/FM metrics are reported to the EMS and NFV-MANO systems.
- Steps 26-27: Based on the **requirements of the Customer**, **Operator analyzes** the operational load, current condition and performance of MF(s), **plans changes to be applied in the MF(s) configuration and the underlying VNF/NS in order to satisfy these requirements**.
- Steps 28-31. **Operator applies the configuration changes** in MF(s) and VNF/NS.

As shown in the list above, making a single MF or a coherent group of MFs (organized into an NS) operational is a complex process that requires a number of concretized decisions to be made by the Operator. This process involves a number of steps executed at the level of multiple management systems and each procedure requires well-structured, concretized information delivered through models and data structures. Obviously, among the presented steps, the majority of the "heavyweight" procedures of Day (-2) and (1) are performed prior the acceptance of NSI/NSSI allocation requests. However, the decisions made by the Operator in Day 0, 1 and 2 can only be made after obtaining the allocation request from the Customer - the Operator analyzes the submitted SLS requirements to be satisfied, the available building blocks, the situation in the operational environment and the business/operational goals to be achieved, and, by taking all these aspects into account, decides on the selection and particular arrangement of the building blocks as well as their appropriate configuration to meet the Customer's request.

2.3. Basic aspects of standards-aligned Network Slicing management

In the previous Section, the management aspects of 5G networks have been discussed from a general perspective, which may or may not involve support for Network Slicing. In this Section, the already introduced information is supplemented with the explanation of the standardized Network Slicing concepts, management models, systems and procedures. Thorough explanation of the utilized notions, relationships between concepts and models, as well as the responsibilities of the management systems is essential for proper understanding of the technological context in which the author's contributions are developed, as well as for their positioning within the standardization landscape.

2.3.1. Basic Network Slicing management concepts

While several notions related to the Network Slicing technology have already been mentioned in the course of this work, their systematic and standardized perception is presented in this Section, supplemented with the description of relevant aspects that have not been discussed yet.

There are four standardized phases in the **lifecycle of an NSI/NSSI object** defined in [11], which are schematically depicted in Figure 14: preparation, commissioning, operation and decommissioning. These lifecycle phases have been defined at the early stages of Network Slicing standardization and remained unchanged since then, without properly reflecting the role of the SLS requirements and their formulation in the NSI/NSSI lifecycle. Hence, while not explicitly assumed in the standards, the author supplements the mentioned phases with the ordering phase, when the NSI/NSSI object does not exist, but activities resulting in the specification of SLS requirements to be met are performed. Specification of such SLS requirements is directly bound to the conceptual, higher-level entity of a Communication Service (CS), which is explained in the upcoming paragraphs.

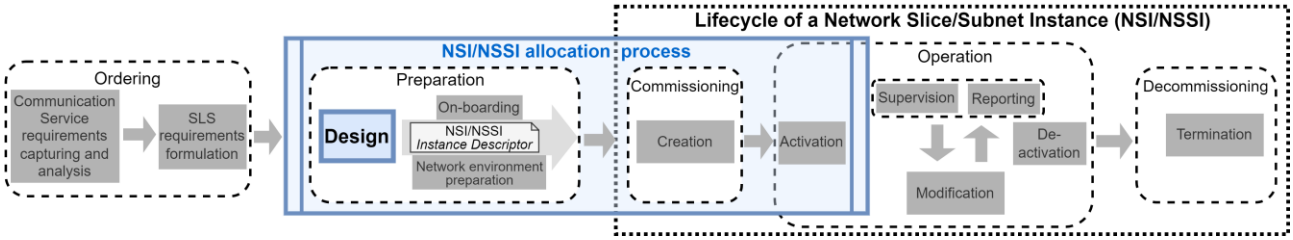


Figure 14. NSI/NSSI lifecycle phases (dashed ovals) and the placement of the NSI/NSSI allocation process within them (blue rectangle).

In the preparation phase (Day -1), the NSI/NSSI object still does not exist. This phase involves the NSI/NSSI **design** based on the SLS requirements to be met (distinguished in Figure 14 with a blue background), on-boarding of its building blocks if not already on-boarded, and pre-provisioning operations that prepare the management systems and administered network environment for the deployment of internals constituting an NSI/NSSI and their configuration[57]. In the commissioning phase (Day 0 and 1), the NSI/NSSI object is created, all resources decided during the design process are allocated and configured for satisfying the SLS requirements. Importantly, the provisioning and configuration of NSI/NSSI and its internals[58] is performed according to the decisions conveyed in an NSI/NSSI *Instance Descriptor* produced as an outcome of the design process.

The operation phase (Day 2) includes the activation, supervision, performance reporting (e.g. KPI monitoring), resource capacity planning, modification and deactivation of the NSI/NSSI object and its internals. Activation makes the NSI/NSSI internal elements ready for handling the Slice traffic. Resource capacity planning is about optimizing the resources utilization of existing NSIs/NSSIs by taking into account the supervision information and performance metrics, which may result in the modification of NSI/NSSI structure. Modification may also be triggered by receiving new sets of SLS requirements to be supported by the already existing NSI/NSSI. Deactivation includes actions that make the NSI/NSSI internal elements stop serving the Slice traffic. The decommissioning phase involves tearing down the non-shared, NSI/NSSI-dedicated internal elements and removing the NSI/NSSI-specific configuration from the shared constituents. After the decommissioning phase, the NSI/NSSI object is terminated and does not exist anymore.

With respect to the lifecycle phases described above, performing NSI/NSSI allocation, semantics of which was introduced in Section 1.2, constitutes a complex process that spans procedures in three NSI/NSSI lifecycle phases, as depicted in Figure 14 with a blue rectangle and as it was depicted in Figure 1 and Figure 2 in Sections 1.2 and 1.3, respectively. In this thesis, the author concentrates on the preparation phase procedures involved in the allocation operation, with a particular focus on the design process. Nevertheless, the other lifecycle phases are discussed as well, because the SLS requirements defined in the ordering phase

[57] These Day -1 pre-provisioning activities may include loading the YANG models into the EMSs, issuing feasibility checks, resources reservations, etc.

[58] Creation of an NSI object may include creation and/or modification of its constituent NSSI objects. In turn, NSSI creation may include creation and/or modification of its internal, nested NSSIs, MFIs and Links.

serve as the most essential input that triggers the design process (discussed in the following Subsection) and the operation phase information about the runtime characteristics of NSIs/NSSIs and their internals, including their performance metrics and faults information (PM/FM data) are taken into account in the design process as well.

A higher-level concept that is frequently utilized in the context of Network Slicing is the notion of a **Communication Service (CS)**, which was introduced in a simplified way already in Chapter 1. In a more normative perception, CS is a broad term used in telecommunications and networking to refer to *any service that enables the exchange of information between users, devices, or systems over a communication network*. There are various types of CSs, including voice services (e.g. traditional telephony, Voice over IP (VOIP), etc.), data services (e.g. access to the Internet, file transfers, web browsing, etc.), video services (e.g. streaming, conferencing, TV signal transmission, etc.), messaging services (e.g. SMS, instant messaging, etc.) and others.

Communication Service Instance (CSI) is defined in [10] as a "*runtime construct of a Communication Service for a defined group of users*". CSIs are offered by Providers to various Customers. Typically, the role of a Provider is played by various **Network OPerators (NOP)** that either own a telecommunication infrastructure (like a Mobile Network Operator (MNO) or Fixed-line Network Operator (FNO)), or do not own it (like MVNO or Fixed-line Virtual Network Operators (FVNO)). In the case of MNOs and MVNOs, CSIs in the pre-5G era have been provided by means of a single network configured in a common, "one-size-fits-all" manner. In the 5G and the future 6G networks, CSIs deployments may be supported with the Network Slicing technology, opening the possibilities of offering the traditional types of CSs with various characteristics and QoS guarantees, as well as providing new types of CSs.

The lifecycle of a CSI consists of 4 phases: preparation, commissioning, operation and decommissioning. While the CSI, NSI and NSSI lifecycles are defined to be largely independent from each other, examples of possible interrelations between them are presented in Figure 15. As shown, the CSI preparation phase, when the `ServiceProfile` containing SLS requirements against the underlying NSI are determined, can be mapped to the ordering phase of an NSI, which precedes its preparation phase. In turn, the ordering phase of an NSSI is triggered in the NSI preparation phase, when the `SliceProfile` is derived from the `ServiceProfile`.

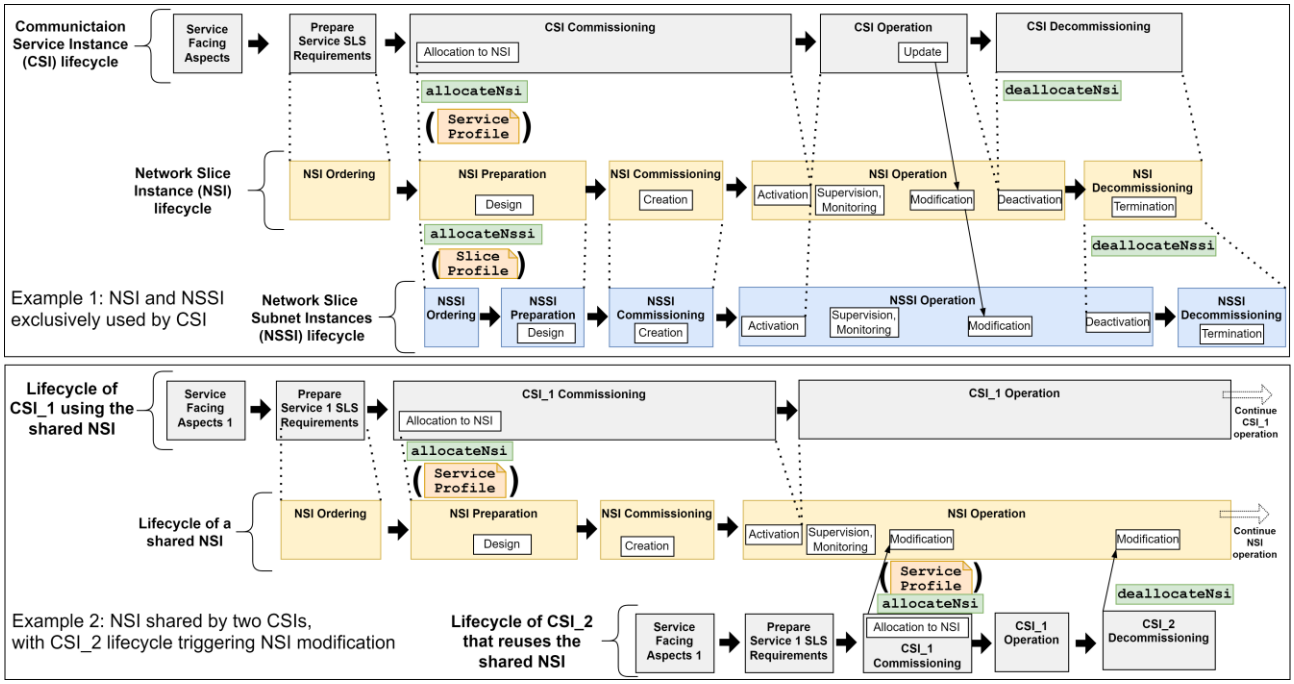


Figure 15. Examples of lifecycle relationships between the CSI, NSI and NSSI objects.

Such a loosely-coupled organization of the CSI, NSI and NSSI lifecycles allows for various operational arrangements of these objects. In particular, the CSI:NSI assignment may be *1-to-0*, *0-to-1*, *1-to-1*, *1-to-1..n*

if more than one NSI is needed to fulfill the CSI's requirements[59] or even n-to-m if multiple CSIs share multiple NSIs. What binds these objects together are the **SLS contracts**, which contain sets of requirements of the higher-level entity against the lower-level one. In the 3GPP terminology, the SLS contracts between CSI and NSI are referred to as **Service Profiles**, since they express the operational requirements of the Service utilizing the CSI. In turn, the SLS contracts between NSI and its NSSIs (as well as between a given NSSI and its nested NSSIs) are called **Slice Profiles**, since they express the requirements of a given Slice imposed on its internal Subnets.

2.3.2. Input triggering NSI/NSSI design and allocation - SLA vs SLS

By utilizing the notions introduced in the previous Subsection, the high-level semantics of the NSI/NSSI allocation operation can be formulated as follows: a given NSI and its underlying NSSIs are provisioned and a share of their resources are devoted to meet the expected operational characteristics of the CSI expressed by means of SLS requirements (Service Profile and Slice Profile respectively), which are derived from the business SLA requirements of a Service that will utilize the CSI. The differences and relationship between SLA and SLS, including explanation why SLS is considered by the 3GPP standards as the main input triggering the NSI/NSSI design process is discussed in the paragraphs below.

The business use cases of the Service envisioned by the Customer and their corresponding requirements imposed on the Telco infrastructure constitute the primary impulse for the allocation of an NSI and its NSSIs. Such business requirements are usually expressed in a form of high-level and, in many cases, abstract **SLA statements** representing the Customer's expectations against the CSI that shall support the execution of the envisioned Service. Examples of SLA statements include: "the CSI needs to be offered in the particular geographical areas" (e.g. a building, district, city, region, country, etc.); "the end-to-end latency should be less than a given threshold"; "the CSI needs to be offered in high-speed trains and in airborne devices up to the altitude of 1km", etc. While mainly focused on the QoS characteristics, the SLA statements may also pose functional or non-functional requirements against the NSI supporting the CSI, like inclusion of a particular functionality (e.g. deep packet inspection), integration with the Customer's systems (e.g. inclusion of its subscribers' database) or security aspects (e.g. particular encryption methods, lawful interception procedures, etc.).

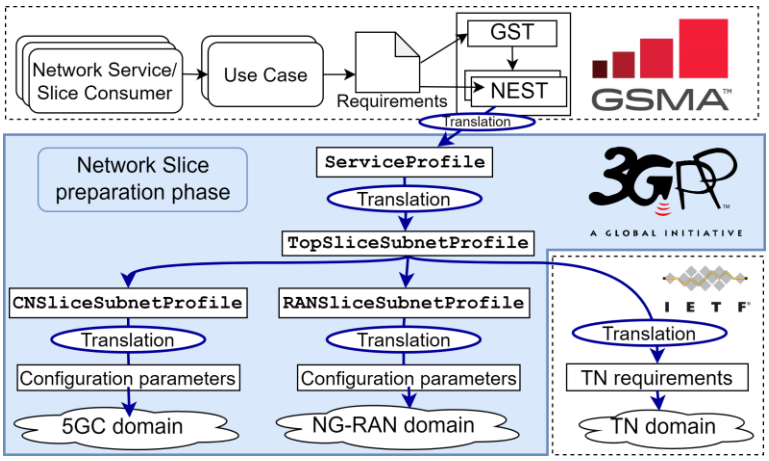


Figure 16. Relation between GST-NEST, `ServiceProfile` and `SliceProfiles` for different domains. Figure based on Annex L of [30] and Annex A of [35].

According to the current standardization consensus [60], standardization of SLA is the responsibility of GSM Association (GSMA) and the standardization of SLS is in the competence of 3GPP, as schematically presented in Figure 16. GSMA defines a Network Slicing SLA template, embodied in so-called Generic Slice Template (GST) [35]. Here, it needs to be underlined that the author finds the name "Generic Slice Template" unfortunate, as it may be confused with the non-standardized concept of a "Network Slice Template" discussed in the upcoming Subsection 2.3.4. Nothing could be more wrong - GST does **not** describe the internal structure of a Slice, nor any of its technical or operational details. Hence, it needs to be stressed that GST does **not** constitute a design-time description of a Slice, which is frequently, wrongly claimed in the state-

[59] e.g. a CSI supporting remote medical consultations [193], [194] or operations might require two NSIs to support immediate interactions and voice transmission (URLLC NSI) and high throughput to transmit large volume DICOM content and high-quality video stream (eMBB NSI). Or, a CSI supporting autonomous vehicles may require two NSIs of types V2X and eMBB in order to guarantee uninterrupted, low-latency communication with the vehicle control systems, and to provide high quality multimedia to the passengers of the vehicle.

[60] This consensus is described in Annex L of 3GPP 5G NRM [32] and Annex A of GSMA NG.116 [35].

of-the-art literature. In fact, GST is just a specification for a set of SLA attributes, which once filled with values, results in the creation of a so-called Network Slice Type (NEST) – again, this name is also unfortunate as it may be confused with the 3GPP-standardized SST[36]. NEST represents an instance of an SLA specification, expressing high-level requirements of the CSI against the envisioned NSI, as stated by the Customer.

The SLA statements, due to their high-level Service perspective and Customer-orientation, are typically too abstract to constitute a sufficiently concretized specification, based on which the technically-oriented process of designing and allocating a suitable NSI/NSSI may be commenced. Hence, according to the aforementioned standardization consensus, the SLA statements in the form of a GST-NEST need to be translated into (or might be supplemented by) a more detailed, lower-level and concretized specification of **technical SLS requirements**, based on which the NSI/NSSI allocation process can be commenced in the management and orchestration systems standardized by 3GPP. As an example, the geographical coverage specified in the GST-NEST's "Area of service" attribute by means of high-level geographical area identification (e.g. name of a country, region, city, building, etc.) should be expressed in the SLS by means of concretized Tracking Area Identity[61] (TAI) list which precisely defines the expected coverage of the NSI/NSSI in the Provider's network.

3GPP specifies the SLS models in the 5G NRM: `ServiceProfile`, as well as several types of `SliceProfile` models containing SLS attributes specific for the particular domains: `TopSliceSubnetProfile`, `CNSliceSubnetProfile` and `RANSliceSubnetProfile`. As already mentioned in Section 1.3, the `ServiceProfile`/`SliceProfile` models **serve as the sole input** [62] to the standardized `allocateNsi`/`allocateNssi` operations, triggering the design and allocation processes in the underlying Orchestrator - automation of these processes is the focus and scope of the author's research described in this thesis. While the author considered inclusion of the SLA→SLS translation in the scope of the research, these aspects have been deliberately excluded for several reasons commented below.

Firstly, the current state of standardization of the GSMA's GST and 3GPP's `ServiceProfile`/`SliceProfile` is immature and not properly harmonized[63]. In particular, due to the lack of comprehensive mapping between these models, the author believes that currently the translation process performed without the involvement of a domain expert may result in inadequate SLS specification due to the generality of the SLA statements and the possibility of their various technical interpretations. Secondly, capturing the SLA requirements and their processing belong to the business aspects of Network Slicing which, although touched upon in this thesis, are not in the focal point of the author's study. For these reasons, in the author's work **it is assumed that the business-oriented SLA requirements against the Slice/Subnet to be delivered have been already captured and translated into the technically-oriented `ServiceProfile`/`SliceProfile` SLS requirements**. Nevertheless, inclusion of the SLA→SLS translation as a part of the elaborated design process framework is not precluded[64]. Hence, it is considered as a future work, along the progressing maturity of the corresponding standards.

Internally, `ServiceProfile`/`SliceProfile` models consist of attributes representing the SLS requirements with specific values to be met by the NSI/NSSI. The standardized set of SLS requirements specified inside of these models is broad, as they contain over 40 attributes - several examples include: `coverageArea`, `dLLatency` (downlink latency upper threshold), `uLLatency` (uplink latency upper threshold), `dLThptPerSlice` (achievable maximum and guaranteed downlink throughput per Slice), `uLThptPerUe` (achievable maximum and guaranteed uplink throughput per UE), `maxNumberofUEs`, `uESpeed` and many more. At this moment it is important to note that supporting even a single SLS attribute is a **technically challenging task**, which requires coordinated development in each layer of the entire Network Slicing stack, both in the functional and management aspects. Hence, a given Network Slicing management solution is unlikely to support all of the standardized SLS requirements from the beginning. Typically, a narrowed subset of the standardized SLS

[61] Tracking Area (TA) is a grouping of Cells. TAI serves as an identifier of a given TA in the Provider's network. See Clause 9.3.3.11 of [195] for details.

[62] This statement is important in the context of related work analysis in Section 2.5, since many of the research works neglect the standards by defining custom APIs and additional inputs, essentially breaking the declarative nature of NSIs/NSSIs provisioning operations enforced by the 3GPP standards.

[63] During the works on this thesis, neither the GSMA's standardization of GST nor the 3GPP standardization of `ServiceProfile` and `SliceProfile` has reached a state that could be considered stable, as with every new release of the GSMA NG.117 [35] and 3GPP 5G NRM [32] specifications their contained attributes change dynamically. Moreover, currently, the discrepancy between the GST attributes and the `ServiceProfile` is large.

[64] Technical realization of such a translation is possible by leveraging the models and methods of the *Requirements decomposition* design activity, which is among the most important procedures of a design process – it is discussed in details in the upcoming Subsections 3.5 and 4.3.4.

attributes (potentially supplemented with non-standardized, vendor-specific ones) is targeted to be supported and gradually expanded with new attributes along the growing maturity of the developed solutions[65]. Actually, such approach is fully standards-aligned, because almost all SLS attributes in the `ServiceProfile`/`SliceProfile` models are specified as optional. For this reason, every Network Slicing Orchestrator should specify which SLS attributes are supported, clearly indicating which of them can be utilized in the invocations of the `allocateNsi`/`allocateNssi` operations by the Customers.

2.3.3. Network Slicing management models

Most of the Network Slicing concepts introduced in the previous Subsections have their corresponding management models defined in 5G NRM [32]. The relationship and cardinalities between these models, including elements comprising a 5G network that have been already introduced in Subsection 2.2.2, are presented in Figure 17 and described below.

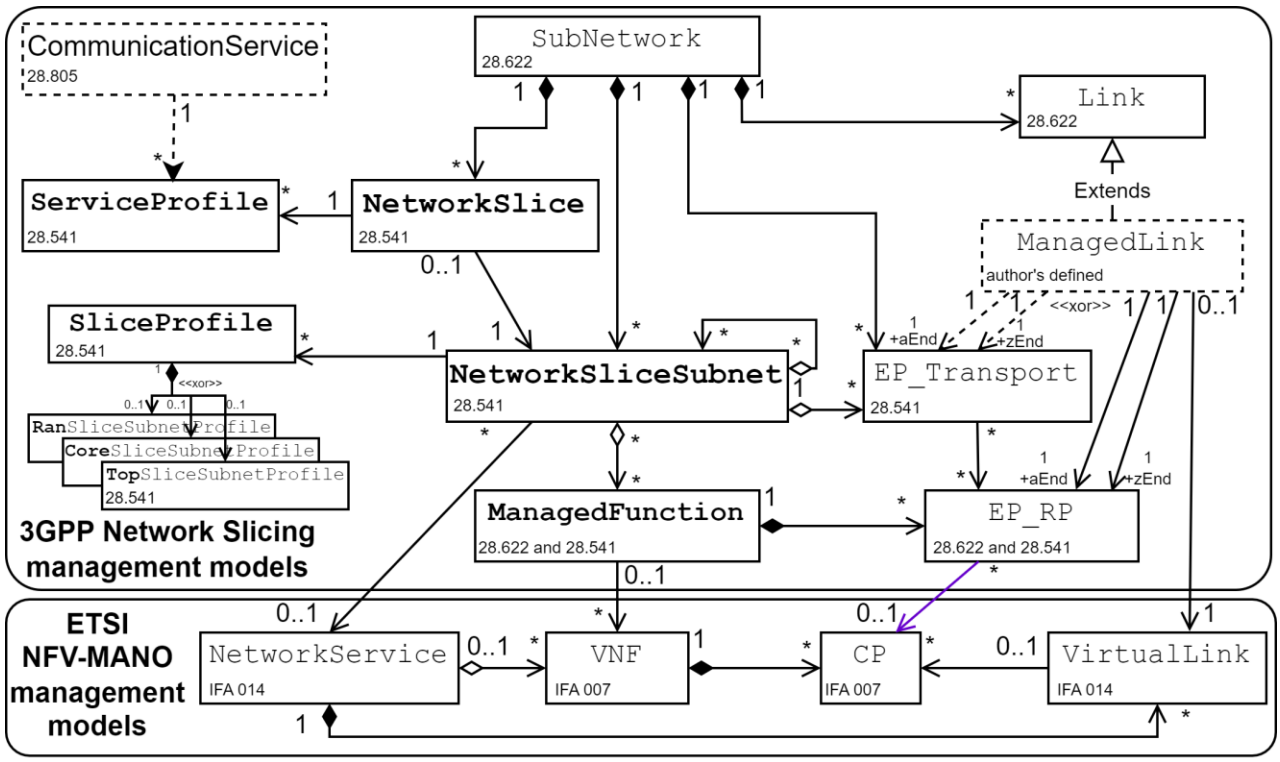


Figure 17. Runtime Network Slicing management models of 5G NRM and their mapping to the ETSI NFV-MANO runtime models.

In runtime, NSIs and NSSIs are represented by instances of the `NetworkSlice` and `NetworkSliceSubnet` models, respectively. Every NSI and NSSI object belongs to a given administrative domain represented by the instance of the `SubNetwork` model, e.g. a particular RAN, CN site or a higher-level domain representing the entire 5G network. Internally, NSSIs consist of instances of various models deriving from the abstract `ManagedFunction` model (MFIs). A given, higher-level NSSI may also contain lower-level, so-called "nested" NSSIs. The MFIs may be sharable between NSSIs or dedicated to operate for a single NSI/NSSI. Individual MFIs and the NSSIs perceived as encapsulated units are connected into a given topology by means of instances of the `ManagedLink` model (MLI)[66]. The SLS contracts at the levels of an NSI and NSSI are represented by instances of the **`ServiceProfile`** and **`SliceProfile`** models [32], respectively.

Another fact that is critical to be noted in Figure 17 is that according to the 3GPP modeling approach there is a **0..1-to-1 association relationship** between the `NetworkSlice` and `NetworkSliceSubnet` models. This means that the concept of an **end-to-end NSI from the management perspective is represented by a distinguished,**

[65] According to [28], the initial rollout of Network Slicing orchestration solutions should support a narrowed set of models, referred to as "lite Slice NRM fragment" and gradually expand the support towards full set of 5G NRM models and attributes.

[66] `ManagedLink` model is an author's defined derivative of the standardized `Link` model [80] which adds essential features currently missing in the standardized `Link` model - possibility of interconnecting NSSIs through their `EP_Transport` endpoints and mapping to the underlying constructs in the NFV layer, like an intra-DC VL or inter-DC MSCS-backed VL. As such, `ManagedLink` provides a single abstraction over any logical connectivity between MFIs and NSSIs, regardless of the underlying technical realization.

"top"[67] **NSSI**. As such, the entire technical information about the internal structure of an NSI (inter alia, information about the MFIs, Links and nested NSSIs it consists of, their configurations, but also their mapping to the underlying NFV layer entities, etc.) is absent in the NSI object itself, as it is conveyed by the referenced "top" NSSI. Although such a modeling approach may seem counterintuitive (because according to the common-sense understanding "an NSI may be composed of many NSSIs"), it allows for **uniform treatment of NSIs and NSSIs at the technical level**. It is because of this fact that the author frequently mentions NSIs and NSSIs jointly, utilizing phrases like "designing NSI/NSSI" or "NSI/NSSI design-time description", etc. Indeed, in the majority of contexts the discussed topics are relevant for both of these entities at the technical level. Nevertheless, it is clearly indicated if a distinction between them is needed.

Another fact to note is that 3GPP describes the concepts related to CS only on the level of an early Technical Report (TR) [10] of informative nature, without covering them in any normative Technical Specification (TS). As such, 3GPP does not provide formalized management models that would directly correspond to the CS concepts. Nevertheless, from the management standpoint, a given **CSI is represented by a particular instance of** `ServiceProfile` that represents the CSI's SLS contract against the NSI. At the level of NSSI, a given CSI is indirectly represented by the instances of `SliceProfile` that are derived from the `ServiceProfile`.

As depicted in the lower layers of Figure 17, Network Slicing models are associated with the management models of the NFV-MANO layer. More specifically, `NetworkSliceSubnet` references a `NetworkService`, meaning that a given NSSI object has an underlying NS instance[68] (if realized in the virtualized infrastructure, which is a default assumption in 5G). In turn, `ManagedFunction` is associated to the underlying one or more VNFs in which it is operating. Finally, the `ManagedLink` model introduced by the author references `VirtualLink` in the NFV-MANO layer. The organization of the 3GPP Network Slicing management models on top of the ETSI NFV-MANO constructs clearly indicates that although the NFV-MANO entities constitute important parts of NSIs/NSSIs, they cannot be considered their equivalents. Thus, management of Network Slicing requires dedicated runtime and design-time models as well as dedicated management systems, which has not always been so clear at the early stages of Network Slicing orchestration solutions development, as discussed in the upcoming Subsection 2.3.5.

All the Network Slicing management models introduced above, together with the models of individual MEs and their layers introduced in Subsection 2.2.2, allow for reflecting the **runtime perspective** of the administered 5G network and the NSIs/NSSIs allocated within it. Based on these models, the duty of the Network Slicing management systems (discussed in the following Subsection) is to create their appropriately configured instances that will result in the **provisioning** of the necessary resources (both "traditional" and the virtualized ones), their operational **configuration** to assure the expected SLS guarantees, and management of these aspects during the lifecycle of the allocated NSIs/NSSIs. However, before the instances of the discussed management models can come into existence, or can be (re-)configured, **their provisioning and allocation needs to be planned**, which requires thoughtful preparation of their design-time descriptions. Unfortunately, similarly as in the case of individual ME models, 3GPP standardization does not provide models expressing the design-time perspective of NSIs/NSSIs which is discussed in the following Subsection.

2.3.4. Lack of standardized Network Slicing design-time models

In Subsection 2.2.3, the design-time models for the individual MEs and their groupings comprising a 5G network have been discussed. As an important observation, it has been pointed out that currently there are no standardized models that would allow for conveying the design-time perspective of the MF applications running inside of MEs and the logical Links binding them. Unfortunately, the same gap is present in the higher layers of Network Slicing modeling. Unlike the NFV layer, where the design-time models are precisely defined by ETSI (see Subsection 2.2.3), **currently 3GPP does not specify any Network Slicing design-time models**. As a consequence, it is currently not commonly specified **what information** should a Slice/Subnet design-time description contain, what **semantics** should it convey, how it should be **structured** and how it shall be

[67] The "top" Subnet is sometimes also referred to as the "root" Subnet. However, "top" is utilized in this thesis to fully align the provided descriptions with the conventions utilized in the 3GPP standards.

[68] This relationship is *n-to-0..1* because there may be multiple NSSIs deployed over the same NS instance (such situation occurs if all MFs inside of the NS are shareable between Slices).

processed. Knowledge of these aspects is essential for elaborating solutions automating the NSI/NSSI allocation process as a whole and, in particular, for automating the design activities embedded in this process. As such, definition of **adequate design-time models equipped with constructs** that will enable supporting *declarative provisioning style* is in the center of the research conducted by the author.

It shall be mentioned, though, that from 2015 (when the NGMN white paper [2] was published) until mid-2019 (when version 16.2.0 of the 3GPP TS 28.531 [95] standard was in force), notions of a **Network Slice Template (NST)** and a **Network Slice Subnet Template (NSST)** were frequently mentioned to serve as Slice/Subnet design-time descriptions, e.g. [96], [97]. However, no standardized specification was ever provided for these concepts [98], [99]. Due to lack of standardization, several incubating Network Slicing projects and research PoCs (e.g. [100] or [101]) attempted to produce their own NST/NSST, specifying their format and semantics according to customized visions. Such custom defined NSTs and NSSTs typically served a similar purpose as Descriptors in the NFV-MANO systems[69], without taking into account the declarative semantics of the standardized NSI/NSSI allocation operations and without considering the **paramount role of SLS requirements** in the design-time description of NSIs/NSSIs. Finally, from mid-2019 (as from release 16.3 of the 3GPP TS 28.531 [102] specification), the notion of an NST and NSST stopped being mentioned in the standards at all, leaving the NST/NSST concepts without clear and commonly acclaimed definition.

Lack of standardized design-time models that would correspond to the 3GPP runtime management models is surprising, because planning the deployment of new MEs and changes in the configuration of existing ones constitute activities that are inherent to the lifecycle of the telco networks in every generation - not only in the preparation (Day 0) and commissioning phases (Day 1) but also during operation (Day 2-n). Perhaps this gap can be explained by the fact that the standardization of the runtime interactions has traditionally been of utmost importance for ensuring interoperability of equipment delivered by different vendors - this factor could have undermined the need of elaborating models conveying the design-time perspective in the 3GPP standardization scope.

The author further recognizes that in the era of statically operated pre-5G networks, provisioning of the "traditional" non-virtualized MEs or changing the configuration of existing ones constituted design-time processes that were performed rarely. Hence, such design-time activities could be carefully planned and performed by Operators in a manual or assisted manner by means of vendor-specific tooling that assisted in making the design decisions. However, while the static, semi-manually performed management by human Operators based on non-standardized design-time models and tools could have been sufficient for the management of the legacy networks, it is largely insufficient in 5G, especially when the Network Slicing features should be supported, and may become entirely unacceptable to operate the future 6G networks.

The author claims that appropriate modeling of the design-time perspective in 5G and 6G is essential for several reasons. Firstly, the management of MFs delivered as VNFs comparing to their PNF counterparts is much more **flexible** and **dynamic**. Secondly, realization of the Network Slicing technology in its full dynamicity requires **frequent** execution of the design activities to reconfigure the existing MFs and Links, set up their new instances or tear down the existing ones. Such need occurs not only when allocating or deallocating NSIs/NSSIs, but also when maintaining and optimizing the QoS/SLS fulfillment of the already running NSIs/NSSIs and their internals as a part of resources capacity planning. Finally, a common expectation is that all these Network Slicing procedures shall be largely or fully automated – this is impossible to be achieved without conveying comprehensive design-time information about entities comprising an NSI/NSSI in models that are **structured appropriately to enable their automated processing** with respect to the standardized, *declarative style* APIs.

Bearing these factors in mind, the author considers the lack of design-time models a serious pain point of the 3GPP standards, which introduces ambiguity in design-time aspects understanding, confusion on how to approach the design-time procedures within the administered 5G network, not to mention automation of the complex Network Slicing design-related procedures. According to the author's firm conviction this results

[69] In the state-of-the-art related to the management and orchestration aspects of telco networks, the notions of a "Descriptor" and "Template" are frequently utilized interchangeably, causing confusion about their semantics. The author specifies precise meanings of these notions in Subsection 3.1.1, as a part of the elaborated *Slicing Entities* taxonomy, which is among the important contributions of this thesis.

in the proliferation of vendor-specific, incompatible solutions and contributes to the slowed-down inception of the Network Slicing technology. Due to the discussed gaps, multiple important architectural decisions need to be made individually when developing a dynamic, automated 5G management solution that supports declarative style APIs. The questions to be answered include, among others:

- Should a design-time description of an individual MF and its interconnecting MLs before instantiation be incorporated into the VNFD/NSD, e.g. by extending the constructs offered by the models of these ETSI-defined Descriptors? Or, shall there be a dedicated, generic model of an MF Descriptor that matches the abstract `ManagedFunction` model defined in the Generic NRM? Or, shall there be dedicated models of Descriptors for each specific MF type to match the 3GPP management models defined in the 5G NRM (e.g. SMF Descriptor, gNB-CU-CP Descriptor, etc.)?
- How to describe the relationship between the MF application's configuration&operational status (e.g. its current and possible to be applied settings, current or predicted load, measured KPI values, time of operation, etc.) and the features of the underlying NFV layer constructs for the fulfillment of the SLS requirements or optimization of MEs operation (e.g. to steer the scaling the underlying VNF, change its DF, migrate, etc.)?
- Should the design-time description of an NSI/NSSI be incorporated into the model of NSD, by extending it? Or should a dedicated Slice/Subnet Descriptor model be established to convey the Network Slicing specifics? If so, how to express the binding of Slice/Subnet management specifics to the features of the underlying NS in the NFV layer?

While the utilization of the NFV-MANO Descriptors and their corresponding constructs for these duties is possible[70], such approach exhibits poor separation of concerns by actually "smuggling" the 3GPP concepts into the NFV-MANO framework which, by design, is unaware of the MF applications and Network Slicing specifics – this distinction becomes even more evident after introducing the standardized functional blocks related to Network Slicing management presented in the next Subsection. Hence, the author's standpoint is that appropriate architectural organization, which assures clear separation of concerns and responsibilities, is to elaborate dedicated Descriptors for Slices, Subnets, Links and the MF application layer that will integrate with the NFV layer by referencing the corresponding NSD/VNFD Descriptors and the constructs they offer. Modeling constructs allowing for building such Descriptors are provided in the upcoming Chapters of this thesis as one of its important contributions.

Moreover, when answering the listed questions and establishing the missing design-time models, two crucial perspectives need to be taken into account, in order to achieve *declarative style* NSI/NSSI allocation: (1) **SLS-orientation of the declarative NSI/NSSI allocation APIs** and central role of SLS requirements fulfillment in Network Slicing in general, and (2) **appropriate preparation** of the elaborated design-time models for **automated processing**. Hence, additional questions need to be answered:

- How to organize the design-time description of Network Slicing management entities (MFs, MLs, Slices, Subnets) to enable automated transformation of the high-level SLS requirements to the low-level configuration items at the level of MFs and their underlying NFV-MANO layer constructs?
- How to supply the instance-specific configuration required for creating instances of MEs that will meet the posed SLS requirements?
- How to deliver various instance-specific configuration for different types of MFs without the need of human Operator intervention?

Importantly, attempting to address these issues by merely gathering the configurable attributes defined in the 3GPP runtime management models and placing them into a structure of Network Slice/Subnet Descriptors inspired by the ETSI NFV-MANO models may not provide the desired added value. Conversely, such structuring would again re-cede the responsibility for making design decisions to yet another higher-level system or a human Operator. To avoid this trap, the design-time models need to be structured in a way that will allow them to **convey the domain knowledge** that is necessary at the particular steps of a design

[70] There have been a number of early attempts of defining Network Slicing management solutions by directly embedding the slicing concepts into the NFV-MANO framework, which is referred to in the description of related work in Section 2.5.

process. The direct presence of this knowledge in the design-time models will free the human Operator from the need of delivering it during the design process and, consequently, **enable automated processing**. Modeling constructs allowing satisfaction of these expectations are proposed by the author in this dissertation.

The gaps discussed in this Section are presented in a broader context in the following Subsection, which discusses the organization of the standardized stack of Network Slicing management and orchestration systems.

2.3.5. Standardized Network Slicing management stack and gaps within it

From the moment of formulating the early Network Slicing concepts at the turn of 2015 [2] and 2016 [1], there have been numerous architectural proposals on how Network Slicing management systems shall be organized, e.g. [87], [100], [103]–[121]. Due to the fact that the ETSI NFV-MANO standards started to be developed earlier, in 2012, natural questions arose in the telco community at that time: could the NFV-MANO components (NFVO, VNFM and VIM) alone be reused "as is" to realize the Slicing concepts? In particular, could the concept of a Network Slice be perceived simply as an equivalent of an NFV-MANO Network Service (NS)? Several solutions (e.g. [87], [109], [113], [122]–[124]) evolved from such an early perception. In such solutions, the NSI object is closely equated with an NS instance and their management blocks are the same. Moreover, tight coupling between NSI and its NSSIs is typically assumed, perceiving NSSIs as NSIs' integral elements, managed by the same NFV-MANO orchestration blocks[71].

However, with the ongoing research and standardization efforts, the distinction between the Network Slicing concepts and the NFV-MANO constructs became sharper. These differences include the already mentioned application-agnosticism of the NFV-MANO entities and, most importantly, central role of SLS management of the MF applications layer in Network Slicing, which are not covered by the ETSI NFV-MANO standardization. In contrast to the aforementioned "NFV-MANO-only approaches", the 3GPP modeling approach described in Subsection 2.3.3 assumes a fully independent lifecycle of the NSI, NSSI and their underlying NS instance objects, shareability of NSIs/NSSIs, their loose coupling by means of SLS contracts as well as their independent management by separated, dedicated orchestration blocks that should expose dedicated allocation and management APIs. These distinctions have been legitimized by 3GPP in the 5G NRM [32] and 28.531 [31] specifications, by defining the Network Slicing management models on top of the ETSI NFV constructs and by defining dedicated management APIs. By taking these facts into account in the standardization works over the years, the most mature and acclaimed architectural framework for Network Slicing management and orchestration has been established by combining the elements of the 3GPP [9], [10], [31], [32], [71], ETSI [81], [84], [85], [88]–[90], GSMA [35] and TM Forum [125], [126] standards, as presented in Figure 18.

The content of Figure 18 puts the most important aspects discussed in the previous Sections into a single view, allowing for clear presentation of standardization gaps (red frames and question marks) and indication of the design-time aspects that are targeted by the author in this thesis (blue oval). The elements comprising the standardized management and orchestration stack, including the standardized management blocks (column C), provisioning APIs (column B) and their operational inputs (column A), as well as the runtime management models (column E) and the models of their design-time description (column D) are commented in the paragraphs below.

As depicted in column C, 3GPP defines three functional blocks responsible for the management of Network Slicing related entities discussed in the previous Subsections: **Communication Service Management Function (CSMF)**, **Network Slice Management Function (NSMF)** and **Network Slice Subnet Management Function (NSSMF)**, which are specified on top of the NFMF, EMS and ETSI NFV-MANO blocks[72] that have already been discussed in Subsection 2.2.4.

[71] Several of the non-standards-compliant approaches, including those equating NSI and NS, are described in more detail in related work, Section 2.5.

[72] Layered organization of the management blocks, as depicted in column C, allows for intuitive presentation of the top-down hierarchy between the functionalities of these blocks. However, due to the adoption of Service-Based Management Architecture (SBMA) in the 3GPP management standardization [71] and advanced steps towards introducing it by ETSI in the NFV-MANO architecture in Release 5 or later [196], [197], this arrangement is purely conventional. It is because the SBMA approach allows for flexible organization of the service-based functional blocks that provide and consume services by communicating over a logical service bus in a peer-to-peer manner, without rigidly specified layers.

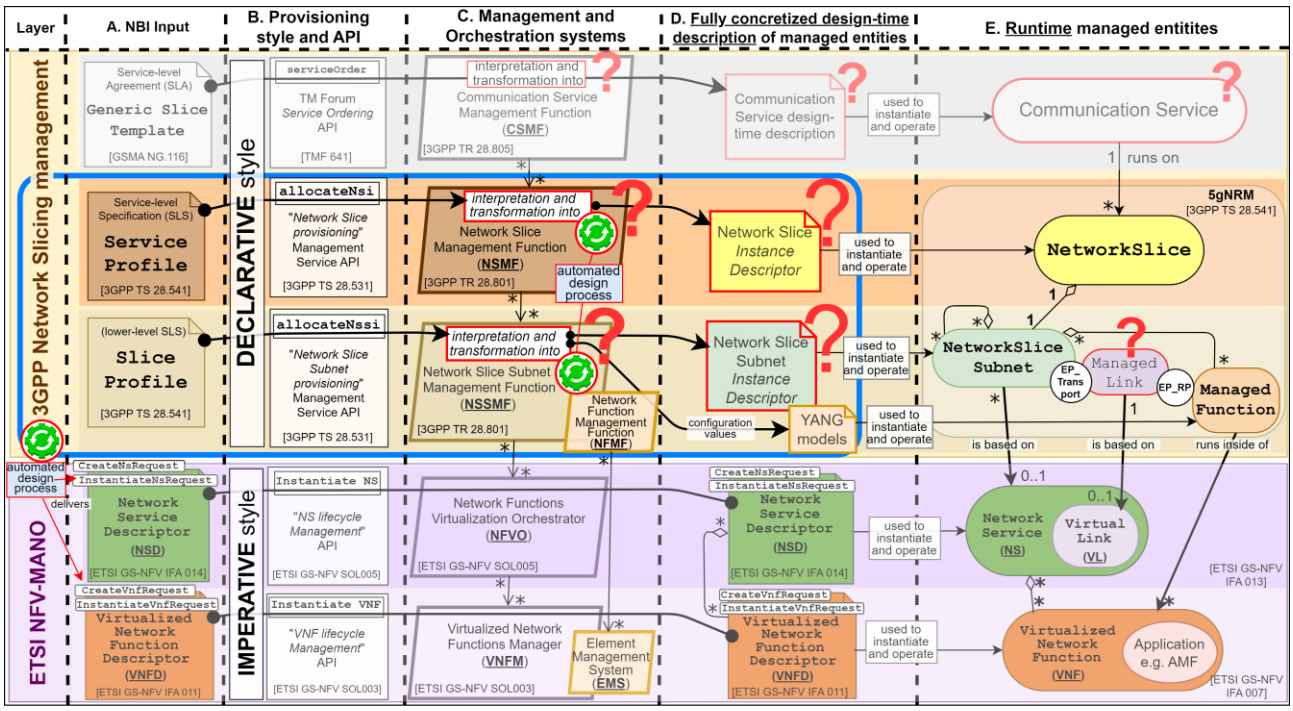


Figure 18. Elements comprising the standardized Network Slicing orchestration stack: management blocks (column C), their northbound interface (NBI) inputs (column A), provisioning style of the corresponding APIs (column B), entities managed by these blocks (column E) and their fully concretized design-time descriptions (column D). Elements with red borders and question marks are not covered by standardization. Blue oval indicates the focus area of the author's work.

The CSMF's main role is to manage the lifecycle of CSIs and their mapping to the underlying NSIs. CSMF is responsible for the Customer-facing aspects of Network Slicing, by obtaining the request for establishing a CSI based on high-level SLA requirements specified by the Customer and translating them into the lower-level, more technically oriented SLS specification in the form of a `ServiceProfile`(s) that can be posed against NSI(s). However, besides specifying the CSMF duties in [9] and [10] in a wishful manner, neither of those 3GPP reports provide details on how to achieve accurate translation from SLA to SLS. Moreover, models and APIs related to CSIs management, both design-time and runtime, are also not specified within 3GPP, which is schematically marked with question marks in the upper part of Figure 18. Although there are attempts from other SDOs to cover some of the corresponding areas[73], the other unspecified aspects mentioned earlier remain challenges to be tackled individually by the implementing parties. While the mentioned CSMF functionalities[74] are relevant for the topics discussed in this thesis, as the output of this block constitutes the basic input triggering the NSI/NSSI allocation, the responsibilities of CSMF are concentrated on the higher-level, business-oriented aspects, which are considered out of scope of the author's research. Nevertheless, generalization of the concepts elaborated in this thesis towards the CSMF level is considered as a future work.

NSMF and the potentially recursively-organized hierarchy of NSSMFs are responsible for the LCM of NSI and NSSI objects, aiming to meet the SLS requirements expressed in the `ServiceProfile` and `SliceProfile` [32] models, respectively. Instances of these models, filled with concretized values of SLS requirements, serve as input for the declarative `allocateNsi`/`allocateNssi` operations [31] that shall be exposed in the NBIs of the standards-aligned NSMF/NSSMF implementations, as depicted in Columns A and B. However, besides high-level sequence diagrams depicting the main allocation steps in a "wishful" manner, **neither this nor any other 3GPP specification provides details on how to achieve the translation of the high-level SLS requirements into the technical specification** of the NSI/NSSI structure of internals and their configuration that would meet them (column C). There is also **no specification of models that would convey the design-time perspective of NSIs/NSSIs nor MFIs or MLIs** (column D). These gaps are marked in Figure 18 with question marks. Elaboration of models and methods allowing to perform an automated design process is the main target of the author's research presented in this dissertation, necessary to verify the stated thesis.

[73] GSMA's GST [35] and TM Forum's generic Service Ordering API [125] are frequently adapted as the NBI of the CSMF block.
[74] A more detailed specification of CSMF's structure and functionalities in accordance with the 3GPP's TR 28.805 [10] is attempted in [198].

By analyzing columns A-C, a clear difference in the style of provisioning APIs and the positioning of the design process is visible between the 3GPP Network Slicing (top rows) and the NFV-MANO blocks (bottom rows). In contrast to the *declarative style* NBIs of the 3GPP blocks that accept high-level SLA/SLS requirements as input and need to perform a design process towards determining a design-time description of manageable entities that will meet these requirements, the *imperative style* NBIs of the NFV-MANO systems need to receive the already fully concretized design-time descriptions of entities to be managed. As such, the NFVO or VNFM blocks do not perform the design process by themselves and the provisioning input serves at the same time as a complete design-time description of entities to be managed (see the lower part of columns A and D in Figure 18). Thus, while in the case of Network Slicing, the design process is expected to be performed automatically inside of the NSMF/NSSMF blocks, the obligation of performing the design process of the NFV entities is ceded to an outside entity – a human Operator, or a higher-level management system like the corresponding NSSMF. This observation is important in the context of elaborating NSI/NSSI design automation solutions, as it makes clear that the fully concretized description of an NSI/NSSI produced as a result of the design process needs to also encompass generation of fully concretized description for the NFV entities that underlie the MEs utilized in the designed NSI/NSSI.

2.3.6. Slice/Subnet offering modes - relation to static and dynamic slicing, provisioning styles and design automation

Currently, two basic **offering modes** of Network Slices are defined by 3GPP in [11] as: "Network Slice as Provider[75] internals" and "Network Slice as a Service" (NSaaS)[76]. Below, the essentials of these modes are described, including how they relate to the *imperative* and *declarative* provisioning styles, what capabilities an Orchestrator needs to offer these modes, and how they influence the need for automation in the design-time phase.

According to the assumptions of the "Network Slice as Provider internals" mode, the Customers utilize CSIs staying unaware of the fact that behind the scenes the Network Slicing technology is supporting their operation in the Provider's network. In this mode, it is expected that a few coarse-grained and large-scale NSIs, comparable to the size of the currently utilized 4G networks are (pre-)provisioned in the Provider's infrastructure, optimized to support a range of typical use cases, mainly within the standardized Slice types (SST[36]). Due to their scale, Slices and Subnets established as "Provider internals" are expected to be shared by multiple CSIs and long-lived – rarely added, deleted or modified.

With such characteristics, offering Slices and Subnets in the "as Provider internals" mode is already possible with *imperative style* Orchestrators that perform so-called "**static slicing**"[77]. Such systems are capable of handling NSIs/NSSIs creation, deletion and modification that happens infrequently and the corresponding resources allocation does not change significantly in time. As Slices/Subnets offered in this mode are not expected to be delivered on demand, the corresponding design activities may be carefully planned and conducted in a highly-controllable manner by the Operator itself in an assisted or even manual manner. Moreover, due to the fact that only a relatively small number of coarse-grained Slices are expected to be created and operated in a rather "static" manner, the design activities need to be performed rarely. As such, investment in full automation of the design process is not necessary to fulfill the business goals of offering Slices/Subnets in the "as Provider internals" mode, at least at the initial stages of Network Slicing commercial rollouts, where a relatively small number of Slices/Subnets "as Provider internals" are anticipated to be operated.

With the growing maturity of the Orchestrators, Providers are expected to offer Slices/Subnets in the NSaaS/NSSaaS mode, allowing the Customers to trigger NSI/NSSI allocation themselves programmatically, by utilizing the `allocateNsi`/`allocateNssi` provisioning operations exposed by the Orchestrators. Due to the

[75] In the 3GPP specifications, the notion of "Network Slice as NOP internals" is utilized instead of "Network Slice as Provider internals". However, for the sake of generality and consistency of nomenclature utilized in the thesis, the latter version is utilized.

[76] Similar modes, although currently not explicitly distinguished in the standards, can be defined for offering Subnets as "Network Slice Subnet as NOP Internals" and "Network Slice Subnet as a Service" (NSSaaS), respectively.

[77] Currently, there is no clear, common, acclaimed nor standardized definition for the terms "static slicing" and "dynamic slicing". A viewpoint on this topic acknowledged by the author is provided in [199].

Customer's requests arriving spontaneously, NSIs/NSSIs offered "as a Service" need to be delivered **on demand**. Moreover, depending on the arrangements between the Provider and the Customer, an agreed dose of management capabilities over the allocated NSI/NSSI may be handed over to the Customer, making an impression of "owning" a fully-fledged Private 5G network. NSIs/NSSIs offered in the NSaaS/NSSaaS mode are expected to be of various **sizes** (from large and middle scale: international, nation-wide or region-wide; to small, lightweight city-wide or even cell-sized), with various **duration of operation** (including short-lived, for the duration of an event like a concert or conference, or only available at periodic timeslots) and instantiated in potentially **massive quantities** (vide the 5G! Pagoda project motto [127]: "A Network Slice for every service!").

To fulfill the aforementioned expectations, Orchestrators need to be equipped with "**dynamic slicing**" capabilities, supporting dynamic NSIs/NSSIs creation, deletion and modification based on the Customers' allocation requests triggered programmatically, which may arrive at unspecified moments in time and in various frequencies. Moreover, with a potentially large count of simultaneously running NSIs/NSSIs, constant adaptation of the existing logical entities and their mapping to the target deployment environments' resources must be performed in order to optimize the utilization of the underlying infrastructure. Achieving "dynamic slicing" entails the need for a high degree of automation in all phases of the NSI/NSSI lifecycle during its preparation, commissioning, operation and decommissioning phases. This need especially concerns the design process, as the NSI/NSSI allocation requests may arrive spontaneously, in potentially large quantities, and the invoking parties expect prompt responses to their requests. To meet these expectations, any operations happening in between the *declarative* APIs invocation and the delivery of a ready-made-NSI/NSSI, including its design, shall be performed in a highly automated way, preferably in an **unattended** manner.

2.4. Requirements to be met by a generic Network Slice/Subnet design automation solution

Bearing in mind the motivations and goals to be achieved in this thesis, as well as the conducted gap analysis of the Network Slicing management standardization landscape, the author searches for a conceptual and practical solution of the Network Slice/Subnet design process automation problem that will meet the following functional and non-functional requirements:

- Functional requirements:
  - programmatic triggering of the NSI/NSSI allocation operation, strictly adhering to the 3GPP standardized APIs [31] and models of the 5G NRM [32]. This entails that the solution must:
    - expose the `allocateNsi`/`allocateNssi` operations that execute based on an input containing solely high-level SLS requirements specified by the Customer in the `ServiceProfile`/`SliceProfile` data structures;
    - not require possessing detailed telco domain&operational knowledge by the Customer;
    - not require human Operator's participation in the design process.
  - hands-free transformation of the input SLS requirements into a ready-made, fully-configured NSI/NSSI that meets the stated SLS requirements;
  - analysis of various building blocks available in the administered domain, both existing that can be shared and the ones to be newly provisioned;
  - easy, declarative way of adjusting the operational variant of the solution to the Operator's operational and Provider's business objectives.
- Non-functional requirements:
  - full alignment to the 3GPP standardized concepts, models, systems and procedures;
  - prompt delivery of design results, supporting on-demand NSI/NSSI allocation;
  - possibility of gradual implementation and extension of the automation scopes;
  - extendibility of the solution to support new operational and Customer's inputs (i.e. new SLS attributes, building blocks, etc.).

Besides setting the direction for the author's research and development, the requirements specified above also served as a reference in the review of the state-of-the-art research papers, which is summarized in the following Section.

2.5. Related work

While the content of this Chapter until now has been concentrated on the analysis of the Network Slicing management and design-time aspects' standardization, this Section provides a critical review of the research approaches that are closely related to the main scope of this thesis. In particular, the most indicative research papers and open-source projects are compared and contrasted against the statement of the declarative Network Slice/Subnet design problem (Subsection 1.3) and the requirements that should be met by a solution to this problem (Subsection 2.4) sought in this dissertation.

2.5.1. Overview of related research

Over the recent years, Network Slicing has gained an enormous attention not only in the telco industry but also in the academic community, resulting in a flood of publications about its various aspects. Regarding the design-time issues, however, most of the state-of-the-art papers either do not discuss them, or treat them superficially, without providing detailed coverage of the design-time challenges pointed out in the previous Sections. In particular, while performing the state-of-the-art analysis of over 100 papers, the author repetitively found statements similar to: *"the Provider receives the allocation request with service requirements and determines a suitable NSI/NSSI that meets them"*, but without providing any details on how it is actually achieved.

Among the papers that do discuss the design-time aspects, the vast majority of publications describe customized, non-standardized approaches towards Slice/Subnet design-time modeling and procedures. Typically, such approaches follow the ***imperative provisioning style***, where the responsibility of devising the NSI/NSSI design is ceded either to the requesting party (Customer) or to the Operator, who manually, or with the aid of assisting tools craft the design of the desired NSI/NSSI, typically in the form of customized design-time models named in various ways: Network Slice Templates, Descriptors, Blueprints, etc. In such approaches, automation of the provisioning and allocation procedures is typically considered only from the commissioning phase, when the fully-concretized NSI/NSSI design-time description devised by the Customer/Operator is delivered to the Orchestrator. Papers with such an orientation provide either high-level [99], [105], [107], [108], [110], [111], [113], [115], [120], [128]–[131] or more detailed descriptions [87], [109], [132], [133] of the design-time aspects. Several indicative *imperative style* approaches are referred to in Subsection 2.5.2.

Treatment of the design-time aspects according to the ***declarative provisioning style*** is signalized by several publications [42], [99], [105], [106], [108], [110], [128], [134], [135]. These papers, however, besides mentioning the need for performing some of the design activities (e.g. SLA/SLS interpretation, dynamic selection of NSI/NSSI internals, determining their dedicated configuration, etc.), do not reveal any details on how to achieve such an automated processing. To date, there is only a handful of recent publications treating the design-time issues according to the *declarative provisioning style* in more details, including [121] (with corresponding papers [112], [136]–[138]) and [8], [139], [140], which mainly revolve around the intent-based management of the telco networks. Selected works describing *declarative style* of provisioning are described in Subsection 2.5.3.

Surprisingly, compliancy of the researched solutions with the 3GPP standardized provisioning APIs and 5G NRM models is attempted **rarely**[78]. A phenomenon that was frequently observed during the state-of-the-art review is that the majority of research works that refer to the standards, do it in a selective manner, complying only to selected aspects (e.g. standardized NSI/NSSI lifecycle phases, standardized business and technical roles, etc.), but usually neglecting the definition of the declarative provisioning APIs and the 5G

[78] Indeed, when searching for the terms "allocateNsi" and "allocateNssi" at the beginning of 2025, both in the popular web searching engines (e.g. Google Scholar) as well as in the major publishers' databases (among others: IEEE eXplore, ACM Digital Library, Wiley Online Libary), the set of research works containing these terms yielded only 7 results, including the author's paper [30].

NRM models, which have been under the standardization process already for a long time[79]. In particular, the research community seems to ignore the fact that the SLS requirements shall serve as the sole input to the NSI/NSSI allocation APIs and there are **no** standardized Network Slicing provisioning APIs that accept NSI/NSSI design-time descriptions as input (e.g. "templates", "descriptors", "blueprints", etc.). In the research works that do consider the standardized models and APIs, their declarative semantics is frequently broken by "smuggling in" various extensions to the `ServiceProfile`/`SliceProfile` models or applying simplifying assumptions, as described in Subsection 2.5.4. The reason for such a state might be that a versatile implementation of these APIs, not limiting the Customers utilizing them and not relying on various simplifying assumptions on the Orchestrator's side, is indeed a complex challenge that requires elaboration of dedicated models and methods to be solved comprehensively.

Although approaches neglecting the standards may have some research value, in the author's opinion, such attempts introduce more confusion to the already complicated Network Slicing design-time aspects. Moreover, in the author's opinion, proliferation of approaches not conforming to the standards moves the telco community farther away from the practical exploitation of this technology, as its utilization in real-life use cases is in fact only possible by its broad commercial adoption. This, in turn, is only possible when the constructed solutions are standards-compliant, which is among the most important aspects in the industrial reality, decisive on solutions' adoption by the Providers.

For the aforementioned reasons, the author distances himself from the "freestyle" research approaches, by placing standards-compliancy as the primary requirement to be met by the elaborated solutions, as well as an important determinant of the potential for industrial exploitation of the research results, which is discussed in the summary of this dissertation, in Subsection 7.2. In this respect, the author's paper [30] and blog publication [141] stand out in the state-of-the-art research, by delivering a comprehensive explanation and coverage for the organization of a generic and flexible Network Slice/Subnet design process that is fully aligned with the standardized APIs [31] and 5G NRM models [32]. The remaining identified research works attempting alignment to the standardized models and APIs are described in Subsection 2.5.4.

2.5.2. Imperative style approaches

Among a number of papers and projects that can be classified as applying the *imperative provisioning style* there are several representative ones that are briefly described in the paragraphs below.

[133] proposes an Orchestrator equipped with a graphical workspace for designing NSIs in a drag-and-drop manner, to be used by both the Operator and the Customers. A so-called Network SLice Descriptor (NSLD) is proposed as a Slice design-time model, constituting a simple overlay on top of the NFV-MANO descriptors (NSDs and VNFDs), supplemented with the claimed specification of SLA and QoS requirements. However, the correspondence between requirements and how are they satisfied by the structure of the designed NSI is not discussed. There is also no dedicated design-time model for Subnets. As such, NSI is de-facto equated with an NS and there is no conformity with the 3GPP notions of NSSIs, Service Profiles or Slice Profiles. MFs sharing among NSI deployments is also not considered - each NSI deployment consists of dedicated MFs only. Surprisingly, although the authors refer to 5G, the particularities of 5G MFs are entirely omitted, as the NSIs are composed of example functions only (authors' implemented firewalls, load balancers or NATs) but not real 5G MFs. Interestingly, [133] is one of the few papers that presents results of the design-time solution evaluation. The authors conducted usability and performance tests, asking a group of 100 people to use their system for designing and deploying 4 NSIs of different complexities. The results show it takes 3 minutes on average for a manual design of an NSI. However, the scale of the designed NSIs in terms of number of MFs, the complexity of their configuration, etc. is not explained. The authors further note that it takes ~30s for the system to accept the design and deliver the NSLD to the Orchestrator. Regarding solution evaluation, this dissertation provides a broad set of results from extensive tests of the implemented design automation solution both from the functional and performance perspectives, which is presented in Chapter 6.

[109] describes the functioning of a proposed, non-3GPP compliant layer of Slicing orchestration blocks on top of NFV-MANO, including "Network Slice Manager" and "SLA Manager". The described NSI design and

[79] The specification of declarative APIs (3GPP TS 28.531) and 5G NRM (3GPP TS 28.541) have been under the standardization process since 2018.

deployment workflow involves multiple actors[80]. Firstly, based on the gained expert knowledge, the "NSD Developer" needs to craft a set of NSDs by taking into account the available VNFDs and the anticipated Customers' needs. Created NSDs are passed to the "NS Provider" who assesses the DFs established in the NSDs and assigns them with SLA Descriptors (SLAD) to mark their SLA/QoS fulfillment potential. Then, the Customer (e.g. representative of a vertical industry) passes to the "NST Developer" the expected KPI values regarding the NSI to be delivered. Then, the duty of the "NST Developer" is to analyze the Customer's KPIs and, based on the available NSDs, their DFs and the assigned SLADs, create an NST that serves as a design-time description of the NSI to be deployed. Again, NST in this solution serves merely as a thin overlay on top of NSDs, just to reference the ones selected by the "NST Developer". Details of configuring the particular MFs that will run inside of the VNFs is not described and shareability of internals is not considered. A ready-made NST is then on-boarded in the NFV-MANO (which proves that in fact it is just another "top NSD" referencing nested NSDs) by the "Slicing Service Provider", who may initiate the NSI deployment based on the NST. The presented design and deployment workflow represents a complicated, purely *imperative style provisioning* of NSIs, in which human Operator's involvement is significant and its expert knowledge is indispensable in the preparations, design and deployment activities.

[87] describes the steps involved in the design and deployment of NSIs composed of non-sharable NFs in a catalog-driven manner. In their approach, the Orchestrator exposes a customized API that gives the Customer the access to a set of predefined "service templates" that contain the specification of the topology of NFs to be deployed, QoS, geolocation and operational requirements to be met by the deployed NSI, potentially leaving various aspects unspecified, to be decided by the Customer[81]. In the described solution, NSI is equated with the NFV-MANO NS, there is no notion of NSSIs and the considered internals are 4G Evolved Packet Core (EPC) MFs only, without taking into account the complexity of 5G Core and RAN MFs. Upon receiving the service order, the Orchestrator constructs so-called Network Slice Instantiation Level (NSL-IL) by identifying which NSD(s) from the catalog need to be used for NS(s) creation and by selecting within each NSD the deployment option (DF, IL) that best matches the features and the performance level required in the "service template". However, no details are revealed on how such intelligence is achieved and how the fulfillment of various requirements is guaranteed by the particular NSL-IL (NSD-DF-IL combination) and its internals, lacking any modeling constructs that would allow SLA/SLS verification at design-time. Furthermore, in [87] there is no mentioning of any kind of requirements decomposition towards the internals building up the NSI. Due to this fact, the structure of the deployed NSIs is fixed in the predefined NSDs[82] and cannot be established dynamically based on the requirements to be met and the capabilities of their fulfillment by internals, which is proposed in this dissertation. The paper also does not address the shareability of NSIs/NSSIs and their internal MFs/MLs do not cover the RAN domain, where the shareability of building blocks and of the medium resources used by them is a complex aspect to be addressed by the design-time solutions. The authors further describe that the Orchestrator *"builds up an NSL descriptor"* to serve as a fully-concretized specification of the NSI to be deployed, including specification of its operation during the entire lifecycle. However, it is not revealed how various elements of such an NSL descriptor are determined (e.g. where do the policy-based workflows come from and how the orchestration actions are derived out of them, how the highly-detailed VNFs configuration settings for the MF application layer are determined, etc.), placing doubts in its automated establishment by the Orchestrator.

There are also several open-source projects that claim support for Network Slicing – such projects may be freely analyzed for the provisioning style, APIs and design-time descriptions used therein. Among them, Open Network Automation Platform (ONAP) [142] stands out with the scope of aspects either already covered or planned to be covered, including an evolving support for Network Slicing. On the roadmap, five anticipated

[80] It seems that the roles of the "NST Developer", "NSD Developer", "NS Provider" and "Slicing Service Provider" can be all merged to a single role of a "human Operator", as referred in this dissertation.

[81] As such, the assumption is that the Customer has telco knowledge and is capable of deciding about various technical aspects of the NSI to be delivered - thus, it seems the proposed system is dedicated to large business tenants searching for "private 5G" solutions rather than small businesses or individuals.

[82] Subsection 3.3.5 discusses the need of generating new, dedicated NSDs as a result of the design process for the NFV-MANO layer, in order to reflect the structure of dynamically designed NSIs/NSSIs. Dynamic generation of the underlying NSDs allows for flexible adjustment of the NSI/NSSI structure to best fit the Customer's SLS requirements, as well as the operational and business objectives of the Operator and Provider.

architectural variants are considered (schematically presented in Figure 19) with various assumptions regarding standards compliancy and the APIs exposed between the orchestration blocks. In the currently pursued scenarios[83], focused on architectural variants no. 1 and no. 4, only the TM Forum-based APIs at the level of ONAP-based CSMF are externally exposed as NBI. These APIs allow CSI ordering by specifying a set of SLA requirements in the invocation payload, alignment of which with the GSMA's GST [35] is anticipated. On the lower, technical levels of NSI/NSSI provisioning, all invocations between the ONAP-based CSMF, NSMF and NSSMF blocks are implemented as internal, cross-workflow[84] calls utilizing custom APIs (marked with red crosses). Currently, there are no explicit plans to provide 3GPP-aligned declarative `allocateNsi`/`allocateNssi` as external NBIs (marked with blue ovals, assumed in variants no. 2, 3 and 5).

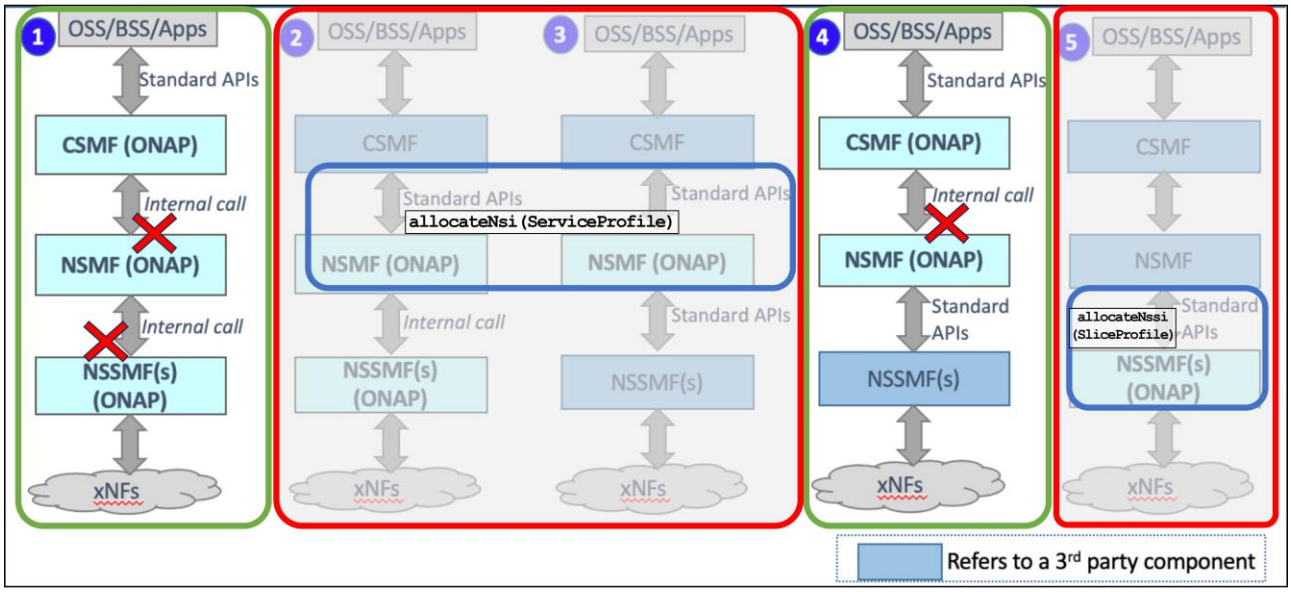


Figure 19. Network Slicing orchestration architectural variants envisioned in the ONAP platform development. Until now, only variants 1 and 4 have been targeted (green ovals). These variants, however, do not assume the exposition of the 3GPP-aligned `allocateNsi`/`allocateNssi` operations in the ONAP NBIs, which is the author's area of interest (blue ovals). Figure based on [143].

Another project built by the open-source community, although at a much smaller scale, is the Open Source MANO (OSM) [123]. OSM, at its current stage of development, offers a simplistic Network Slicing orchestration approach that is not aligned with the concepts standardized by 3GPP, e.g. there is no notion of the SLS requirements and their dedicated management. Moreover, the NSI objects, instead of being detached from their underlying NSs, are perceived as their equivalents. These simplifying assumptions resulted in a purely *imperative provisioning style* by means of customized Network Slicing management APIs established based on the ETSI SOL005 [89] specification, which is natively used for NSs management. As such NSIs are treated similarly to NSs and their design-time description called "Network Slice Template" (NST) plays the role of a simple overlay on top of NSDs that describe the Subnets building up the Slice (strangely, there is no notion of an NSST). The OSM provisioning APIs do not provide any design process automation, requiring the Operator/Customer to provide a fully concretized NST and NSDs to the provisioning APIs. There is no evidence of any plans to implement the *declarative style* of provisioning or align the provisioning APIs and models to those standardized by 3GPP.

2.5.3. Declarative style approaches

Among the papers that can be classified as describing the *declarative provisioning style*, there is one of the first publications [8] related to the Network Slicing technology in general, published already in 2016, before any works related to the standardization of the management and orchestration aspects have been commenced[79]. In this paper, the authors introduce the NSaaS concept, pointing out that its technical realization needs the *"feature of mapping service demands **automatically** from a customer to functionalities, topology, policies and parameters of a network slice, as well as providing component-based and **auto-***

[83] Targeted in the current "Oslo" [143] as well as in the previous releases, beginning with "Frankfurt".
[84] ONAP utilizes a Business Process Model and Notation (BPMN) engine.

configured *network functionalities for operators to* ***design*** *and launch network services more conveniently"*. To achieve that, the authors suggest defining three levels of conceptual management models: application level, network function level and infrastructure level models. Importantly, the authors underline that the application-level models *"should be easy to understand even for application developers without any telecommunication background"*. Further, they describe that *"important description of the application requirements is the SLA"*, pointing out that *"customized SLA requirements [..] must be translated from Customers' orders"*. As it turned out, many of the concepts described in this paper found their embodiment in the 3GPP standards, as described in the previous Sections. In particular, this publication might have influenced the definition of the declarative style provisioning APIs, which are based purely on SLS requirements against the telco infrastructure that are derived from the Service SLA requirements.

The authors of [8] further describe the conceptual approach towards automated slice delivery based on the Customer's order, to be achieved by a dynamic mapping of SLAs at the aforementioned application, network functions and infrastructure levels, performed against the vendor-specific elements with registered functionalities and capabilities. The need of reflecting the abstract, application-level SLA into the SLAs on the lower-levels[85] is underlined. The authors also point to the challenges of automating the delivery of detailed configurations to the vendor-specific elements to guarantee requirements fulfillment. However, besides a general view of the needed orchestration functionalities, the paper does not provide details about the actual way of achieving them. In this regard, the research presented in this dissertation addresses all of the mentioned concerns, providing a holistic and comprehensive elaboration of design-time concepts, models and methods that allow practical implementation of a fully automated, flexible and dynamic design of NSIs/NSSIs based on the high-level SLS requirements specified by the Customer.

The *declarative provisioning style* also encompasses the approaches within the recently observed trend towards **intent**-based management of the telco networks, surveyed among others in [144], [145]–[147]. In such an approach, the requesting party specifies its expectations against an object to be provisioned and managed (e.g. a single NF, or an NSI/NSSI as a whole) in the form of declarative statements regarding its characteristics or desired state. Such declarative statements are referred to as "intents" and are typically expressed utilizing various domain specific languages that aim at mimicking a natural language (e.g. Nile [148]) or via Graphical User Interfaces (GUI). Depending on the advancement of the intent-based system's implementation (among others, the features of its interpreter [149]), an intent may express high-level, Customer-perspective requirements to be satisfied by the managed object, and/or it may target lower-level details of its structure, relevant for the Operator, e.g. expect certain topology, utilization of specific functions with particular configurations, protocols, etc.

The intent-based NSIs provisioning is described, among others, in a series of similar papers [112], [136]–[138] by the same group of researchers, with the broadest coverage of results described in [121]. The authors propose an orchestration system that *"takes higher-level QoS requirements from the customers/operators for a slice and translates them into specific slice template/policy configurations acceptable by the underlying Orchestrators"*. The supported intents constitute QoS requirements passed by the Operator/Customer by means of a GUI which behind the scenes triggers an *"Intent Manager"* that is claimed to be capable of automatically determining end-to-end slice design based on such an input. However, although promising high flexibility of expression and interpretation of intents, the authors do not provide their concretized examples, nor any precise models and intelligence behind the algorithms that are claimed to resolve the intents towards the fully-concretized descriptions of the NSI and its internals. The content of the mentioned papers only reveals that the solution assumes predefined intents represented internally as "chaining graph templates", which are somehow mapped to the particular VNFs and their configurations. The intent layer of the proposed system is defined directly above the OSM NFV-MANO orchestrator. Such organization of the system by the authors effectively equates the NSI with an NS, neglecting the separateness of these concepts defined by the 3GPP standards and the need of NSIs/NSSIs management by dedicated NSMF/NSSMF blocks (see Subsection

[85] This important feature is systematized by the author as the *Requirements decomposition* activity. Concretized models that enable conveying the expert knowledge needed for performing this activity (see *Requirement Decomposition Rules* (RDR) in Subsection 3.5.1) as well as methods of processing these models (see Subsection 4.3.4) are proposed and experimentally evaluated as an important contribution of this dissertation.

2.3.5). Internally, the "NSIs" designed and deployed by the authors' system consist of 4G NFs, assuming non-shared deployments of the 4G Evolved Packet Core NFs per each NSI.

Another example of the intent-based approach towards NSIs provisioning is described in [140]. The authors propose a system supporting the establishment of NSIs based on intents expressed by the Nile language with custom extensions. In their solution, the intents contain a mix of high-level (e.g. throughput or QoS profile requirements) and lower-level (e.g. topology, cardinality of NFs, etc.) expectations against the NSI and its internals to be deployed. A system component referred to as *"Intent Engine"* parses the input intents and extracts the values of the particular statements, checking the feasibility of their realization in the managed environment, but without precisely explaining how this is achieved. The *"Translation&Execution"* component performs a simple mapping of the extracted values towards the particular attributes and configurations of the VNFDs (their identity) and NSDs (topology, VNFs cardinality) that are utilized for the deployment of an NSI. As such, the authors of this paper again equate NSI with an NFV-MANO NS, neglecting the 3GPP-defined orchestration blocks, 5G NRM models and provisioning APIs.

In this context, it is worth mentioning that 3GPP started to follow the intent-based trend from late 2020, kicking off a related standardization process. In particular, [147] specifies the concepts of Customer, Provider and low-level NOP-level intents. The same specification mentions the possibility of their utilization for slicing-related aspects, but without providing details on how to implement them. Actually, the SLS requirements to be satisfied by the designed NSI/NSSI can be perceived as a Customer-level subset of intents' targets. With such a perception, the `allocateNsi`/`allocateNssi` operations can be viewed as intent-based, as they accept and process Customers' intents expressed as high-level SLS statements contained in the `ServiceProfile`/ `SliceProfile` models. Generalization of the concepts elaborated by the author towards supporting intents at other levels of abstraction is considered as a future work, which is discussed in the upcoming Section 7.4.

2.5.4. Approaches attempting standards alignment

[28] provides a neat status of Network Slicing technological rollout, including a high-level description of the NSI allocation procedures based on standardized APIs and 5G NRM models. For the initial rollouts, the authors suggest the implementation of the `allocateNsi` operation based on a so-called *"Baseline Decision Engine"*, which follows predefined mapping between a limited number of GST-NESTs mapped to `ServiceProfiles` and statically defined NST/NSSTs, which in turn, are tightly coupled with the particularly configured NSDs in the NFV-MANO layer. Such approach can be classified as *imperative style provisioning*, because all the design decisions are predefined manually by the Operator and deposited in a tightly-coupled structure of the enumerated, interrelated design-time descriptions, making the structure provisioned NSIs static and rigid. SLS fulfillment feasibility checks are mentioned without specifying how to devise the lower-level configurations of NSI/NSSI internals from the `ServiceProfile`/`SliceProfile` SLS requirements, suggesting to bet on trial-and-error approaches. Shareability and current load of NSI building blocks with the already assigned SLS requirements are not considered.

While the descriptions in [28] focus on the "as is" rollout status, regarding the future, the authors correctly note that with many more NSIs/NSSIs running in parallel, each with completely different requirements, Orchestrators must become much more **dynamic**. For achieving such dynamicity, the authors predict the utilization of so-called *"Advanced Decision Engine"*, but without providing details of how to achieve its anticipated functionalities. The authors also correctly note the need for a gradual expansion of coverage for the management models contained the 5G NRM, especially regarding the gradual support of SLS attributes specified in the `ServiceProfile`/`SliceProfile` models (which is also pointed out in this dissertation in Subsection 2.3.2). In general, the paper provides high-level descriptions, without digging into the details of the specific procedures that take place in the Network Slice/Subnet design process. Referring to this publication, **the solutions elaborated in this dissertation can be perceived as the realization of the *"Decision Engine"* in its *"Advanced"* variant**.

A simplistic implementation of the `allocateNssi` operation is employed in [124], where the authors overcome the problem of automated NSSI design by "smuggling in" an explicit NSD ID and DF ID inside of a set of predefined `SliceProfiles` that are pre-registered in the NSSMF system. Such approach breaks the

standards, because the `SliceProfile` model shall be devoted to convey the SLS requirements only. It also limits the Customer's flexibility of requesting NSSIs allocation with specific characteristics, by requiring the Customer to select from a few pre-registered `SliceProfiles` only. Finally, such approach is against the declarative nature of the operations defined by 3GPP, ceding the responsibility for making the design decisions by the Operator in an *imperative style* before obtaining the `allocateNssi` requests.

Another paper that mentions the utilization of the standardized `allocateNsi`/`allocateNssi` operations for NSI/NSSI provisioning is the conceptual work regarding Network Slicing support in satellite-based networks [150]. In particular, the authors mention the existence of a conceptual *"Slice Designer"* component, the role of which would be to *"design the optimal slice in an automated manner according to the parameters received from the slice manager in the slice creation request"*. The authors further note that *"the Slice Designer holds the intelligence to translate the generic requirements into understandable parameters"*. At the same time, the authors note that "***such algorithms do not exist today and need to be defined***". In this respect, this dissertation is devoted to addressing exactly the mentioned aspects.

[151] refers to the standardized 5G NRM models and APIs, describing the high-level logic behind the `allocateNsi`(`ServiceProfile`) execution. In the approach describe therein, the NSMF follows static mappings between the input `ServiceProfile.sST` field and predefined NSTs that describe the structure of Slices of the supported SST[36] types. Each NST is statically mapped to a set of NSSTs, which in turn are statically mapped to the particular NSDs in the NFV-MANO layer, defining a priori a rigid structure of the Slice and its needed Subnets. The authors claim that by following such mappings the *"slice decomposition"* takes place and the `SliceProfiles` for the particular domains are generated, but without revealing any details on how it is achieved. With this respect, this dissertation proposes clear modeling constructs that allow for achieving such functionality in a flexible manner[85]. Worth noting, [151] correctly points that both NSIs and NSSIs can be shared and, as such, assigned with multiple `ServiceProfiles`/`SliceProfiles`. Thus, NSIs/NSSIs and their internals need to be checked for feasibility of handling additional requirements. However, the way of performing such checks is not detailed. In this regard, this dissertation proposes clear modeling constructs that enable exposition of the current and/or declared SLS fulfillment of the building blocks out of which an NSI/NSSI can be built[86], as well as handling multiple `ServiceProfiles`/`SliceProfiles` assigned to them.

[151] further describes the processing behind the `allocateNssi`(`SliceProfile`) call as *"NSSMF works with the optimization engine and instantiates a new NSSI for that SliceProfile"* without providing any details on how it is achieved. Presumably, the previously determined NSSTs (from the static mappings) are bypassed in the `allocateNssi` call, which not only breaks the declarative nature of the standardized provisioning APIs but also mixes the responsibilities of the NSMF and NSSMF blocks[87]. The authors of [151] also do not reveal how the detailed configuration of NSs, VNFs and MFs to fulfill the particular SLS requirements is performed – presumably all these important details are hardcoded in the predefined SST→NST→NSSTs→NSDs mappings chain mentioned in the previous paragraph. Contrasting with such approach, the solutions elaborated in this dissertation allow for flexible composition of NSI/NSSI structure based on dynamic requirements-capabilities matching, without assuming any rigid, defined a-priori static mappings.

The authors in [152], although meticulously referring to the standardized provisioning APIs and models, only focus on extending the semantics of one of the SLS attributes related to the shareability of NSIs that is defined in the `ServiceProfile` model. While concentrating on proposing means for supporting this single aspect in the `allocateNsi` requests, the authors do not provide any details of a general design process that would interpret the entire input `ServiceProfile` and design an NSI capable of fulfilling the SLS requirements contained therein. In turn, [153] mentions the utilization of the `allocateNssi` operation, but assuming that *"the end-user* (Customer) *specifies the* `ServiceProfile` *which contains slice characteristics such as the S-NSSAI*[88]*, default resource configuration and target NFs".* Again, with such a wrong assumption, the authors

[86] See the *Slicing Entities* taxonomy and the *Slicing Entity Capabilities* modeling construct described in Subsections 3.1.1 and 3.1.2, respectively.

[87] The authors in [151] assume the existence of a *"Flow Engine"* component that collectively realizes the functionalities of the CSMF, NSMF and NSSMF, mixing their responsibilities inside of a single component. This means that the management of the CSIs, NSIs and NSSIs is performed by a single orchestration block, without proper separation of concerns and APIs.

[88] S-NSSAI (Single Network Slice Selection Assistance Information) is a Slice identifier consisting of SST and Slice Differentiator (SD), which differentiates Slices within a single SST type.

break the declarativeness of the `allocateNssi` operation, claiming that the input `SliceProfile` contains a concretized technical specification of internals that should comprise the NSSI to be allocated. Such approach cedes the responsibility for devising the NSSI specification to the party invoking the `allocateNssi` operation, while it should be the NSSMF block that resolves the input SLS requirements to a design-time description of a suitable NSSI.

A recent literature review presented in [98] recognizes that the interpretation of high-level SLS requirements in `ServiceProfile` and the derivation of a design-time description of an NSI that would meet such requirements remain valid problems that currently do not have comprehensive solutions. Among others, this report searches for answers to the following question: *"What are the standard methods/approaches for creating or designing a 5G network slice template*[89]*?"*. The authors confirm that currently there is no standardized specification of any "Network Slice Template", noting that the corresponding research is vague. The authors only enumerate the attributes contained in the customized visions of NST proposed in [108] and [154], without providing any details on how a design process should look like. In fact, despite claiming that the authors searched for *"standardized methods/approaches"*, the works they reference do not utilize the standardized provisioning operations nor the standardized 5G NRM models. Instead, the referenced approaches again represent customized, *imperative style* of provisioning that is based on static, predefined NSI/NSSI design-time descriptions passed as input to the APIs of the proposed orchestration frameworks, without acknowledging the standardized APIs in which the SLS requirements shall be the sole input for the provisioning operations.

The content of the mentioned study [98], as well as the observed abundance of interchangeably utilized design-time notions with various meanings (e.g. "descriptors", "templates", "blueprints", "decomposition", "SLS fulfillment", etc.) in the analyzed papers confirm that currently there is a large doze of confusion around the Network Slicing design-time aspects and lack of their common understanding. The literature analysis also shows a very limited adoption of the Network Slicing provisioning standards in the state-of-the-art papers, as well as their various and frequently wrong understanding. The author of this thesis believes that his research described in [30] and, in a broader scope, in this dissertation, contributes to the demystification of these aspects, promoting clear, fully standards-aligned systematization of the Network Slicing design-time models and procedures. To the best of the author's knowledge, to date, this dissertation and [30] are the broadest and most comprehensive elaborations of the standards-aligned Network Slicing design-time aspects currently published. Comparing to [30], this dissertation contains multiple conceptual refinements (described in Chapters 3 and 4), detailed description of their practical implementation (described in Chapter 5), as well as functional validation and performance evaluation of the constructed PoC solution (Chapter 6).

2.6. Summary

This Chapter provided a detailed walkthrough of the state-of-the-art related to Network Slicing management in 5G networks, with a particular focus on explaining the standardized models, systems and provisioning procedures. The walkthrough has been conducted in a gap analysis manner, which allowed for identifying the key standardization deficiencies that make it unknown how to approach practical realization of the *declarative style* NSI/NSSI allocation operations standardized by 3GPP. The most significant gaps include: lack of MF, ML, Slice, and Subnet design-time models, as well as lack of clearly defined design-time procedures that need to be executed as a part of the NSI/NSSI allocation logic. As a result, it is currently not clearly specified how to transform the high-level SLS requirements into a technical specification of an NSI/NSSI capable of meeting them. Filling-in the identified gaps with concepts that will enable automated, declarative NSI/NSSI design and allocation in a fully standards-aligned manner is challenging and constitutes the main motivation of the research performed by the author.

The author underlined the need of elaborating the missing design-time models by considering two crucial aspects: (1) central role of SLS requirements fulfillment by the building blocks out of which the NSI/NSSI is designed, and (2) appropriate construction of the design-time models that would enable hands-free

[89] The authors of [98] utilize a confusing notion of a "Network Slice Template" that is supposed to represent a fully concretized design-time description of an NSI. Due to the various interpretations of notions like "Descriptor", "Template", "Blueprint", their semantics is attempted to be systematized in this thesis, as a part of the *Slicing Entities* taxonomy elaborated in Subsection 3.1.1.

processing of their instances in an automated design process. To address these concerns, it is essential that the proposed models convey the necessary operational and domain knowledge, possible to be leveraged in the logic of automated design-time procedures. Furthermore, the organization and precise processing logic of the design-time procedures also need to be proposed, as currently it is not specified by any standard specifications.

Besides detailed elaboration of the Network Slicing management standards, also the current research initiatives in the NSI/NSSI design and allocation automation area have been analyzed. It has been pointed that the majority of the state-of-the-art research has been concentrated on *imperative style* approaches towards NSIs/NSSIs allocation and management, which, due to the need of conducting the design process manually, are unable to satisfy the dynamic, flexible and automated allocation of NSIs/NSSIs targeted to be achieved in this thesis. While several *declarative style* approaches have been emerging recently (mainly concentrated around the intent-based concept), support of the standardized management models and APIs has been attempted rarely, in limited scopes or with non-standardized semantics. Against this background, the originality and significance of the author's work were outlined.

Chapter 3

Network Slicing design-time concepts, definitions and modeling constructs

Before any complex process can be constructed and automated, it is crucial to perform its detailed analysis in order to gain a thorough understanding of its inputs, outputs and the processing logic it should execute. To this end, the current Chapter and the upcoming Chapter 4 is devoted to breaking down the Network Slice/Subnet design automation problem into graspable pieces that can be analyzed individually, **with the aim of proposing concepts enabling automation of the particular design activities and the design process as a whole**. In the current Chapter, the focus is on the identification and systematization of **basic inputs** to the design process and its **expected output**. During the analysis, the author introduces and explains several fundamental concepts and definitions that not only constitute essential foundations for constructing automation solutions satisfying the requirements stated in Section 2.4, but also fill the current void in the state-of-the-art and standardization regarding the Network Slicing design-time aspects.

Firstly, in Section 3.1, the building blocks out of which a Network Slice/Subnet may be designed are classified and modeling constructs enabling their automated processing are proposed. Next, in Section 3.2, generic definitions of the Network Slice/Subnet design process and its expected outcomes are provided. Then, in Section 3.3, the structure of standards-compliant Slice/Subnet *Instance Descriptors* is proposed, based on a comprehensive analysis of information items it should contain. Lastly, in Sections 3.4, 3.5 and 3.6, a range of modeling constructs that enable capturing the necessary domain&operational knowledge and its exploitation for enabling automated execution of individual design activities are provided. The concepts proposed herein fulfill the dissertation goals in subsequent Chapters, which present their utilization in a generic conceptual framework of an automated Network Slice/Subnet design process (Chapter 4), as well as their practical embodiment in a PoC implementation (Chapter 5).

It needs to be noted that, in their original form, the core concepts discussed in the current and the upcoming Chapter 4 have been submitted for patenting [45], [46] and then presented in the author's major publication entitled *"Comprehensive tutorial on the organization of a standards-aligned Network Slice/Subnet design process and opportunities for its automation"* [30]. In this dissertation, the content of this publication is frequently adopted, including refinements introduced based on further research iterations and experience gained from the practical implementation of the elaborated concepts (described in Chapters 5 and 6).

3.1. Modeling of Slice/Subnet building blocks – Slicing Entities

Before diving into the course of the design process and the outcomes it shall produce, it is crucial to understand the **nature of entities** that are considered for designing a Network Slice/Subnet. The author's thorough investigation of thereof allows for introducing a fundamental notion of a ***Slicing Entity***, defined as a *logically encapsulated Network Slice/Subnet building block that exposes its* ***Capabilities***. Due to its crucial meaning in the analysis of the design process and in the elaboration of modeling constructs enabling its automation, the notion of a *Slicing Entity* and its related concepts are thoroughly explained in this Section.

Firstly, in Subsection 3.1, a **taxonomy of *Slicing Entities*** is provided, which classifies their large heterogeneity according to several criteria that are important in the context of the design process and its automation. After discussing the ways in which *Slicing Entities* differ, common characteristics of *Slicing Entities* embodied in the crucial concept of *Slicing Entity Capabilities* are explained in Subsection 3.1.2. Then, in Subsection 3.1.3, the

modeling construct of *Slicing Entity Configuration Profile* is introduced, which is especially important for design automation. Finally, in Subsection 3.1.4, interpretation of *Capabilities*' values, which needs to be taken into account in the design process, is discussed.

3.1.1. Slicing Entities taxonomy

Figure 20 depicts the taxonomy of *Slicing Entities*, by classifying them according to six criteria: (1) represented lifecycle stage, (2) abstraction level, (3) concretization level, (4) class of represented entity, (5) utilized models, and (6) the type of entity within a given class. Importantly, a given *Slicing Entity* is an identifiable specimen of an entity (an Instance) present in the Orchestrator's inventories during operation, **not** its model.

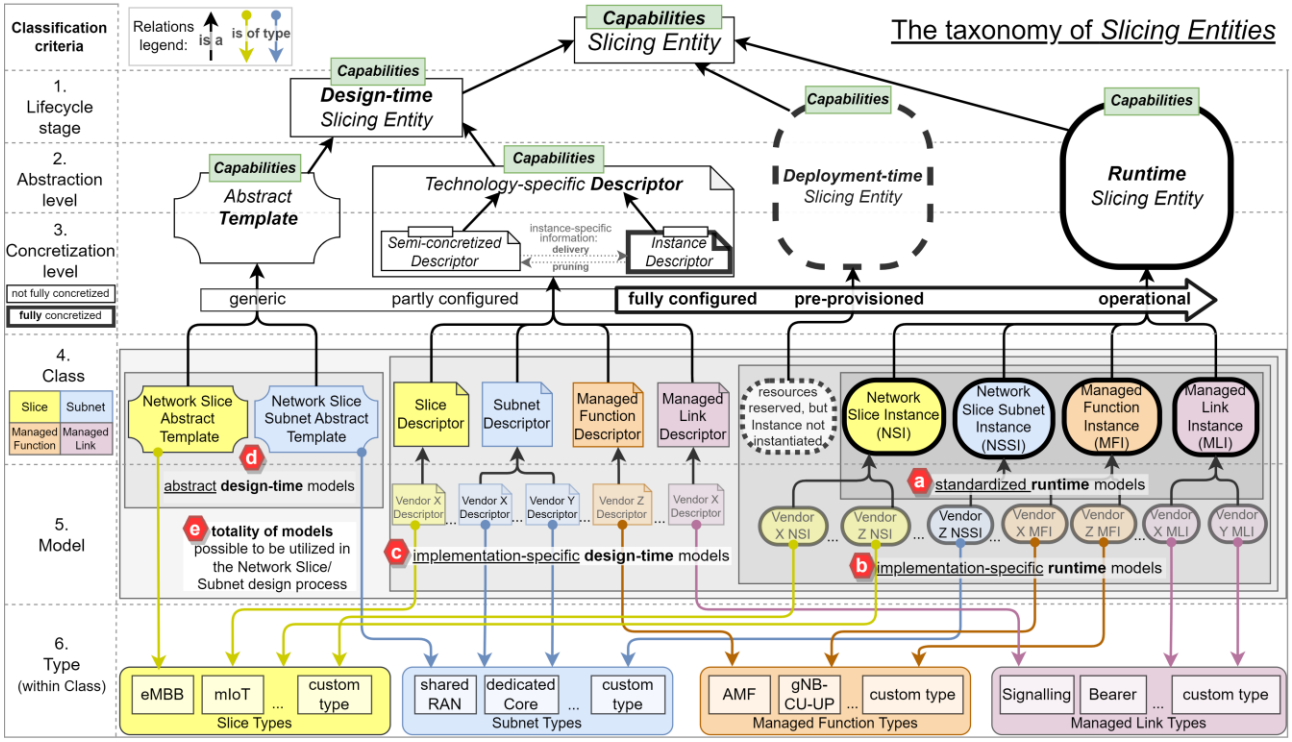


Figure 20. The taxonomy of *Slicing Entities* according to six classification criteria: (1) lifecycle stage, (2) abstraction level, (3) concretization level, (4) class, (5) model and (6) type within the class.

The first classification criterion of *Slicing Entities* is the (1) **lifecycle stage** they represent. According to the basic Network Slicing principles, it must be possible to design a Network Slice/Subnet out of instances already existing in the administered telco network, as well as from instances that will be newly spawned (based of their design-time descriptions). Addressing this requirement results in the basic division of *Slicing Entities* into: ***Runtime*** *Slicing Entities*, typically represented as **objects** inside of the operational inventories of the Network Slicing orchestration blocks (NSMFs, NSSMFs, NFMFs), and ***Design-time*** *Slicing Entities*, typically represented as **files** stored in the catalogs of such Orchestrators[90]. This basic division can be supplemented by *Slicing Entities* representing an intermediate, ***Deployment-time*** lifecycle stage, usually observed in the case of entities for which resources have already been reserved in the target deployment environments, but no actual instances have been created yet[91]. Such entities are included in the taxonomy, because if a given *Deployment-time Slicing Entity* is shareable, then it is permissible to consider it for inclusion in the currently designed Slice/Subnet, besides the ones that have already reserved its usage.

The second classification criterion is the (2) **abstraction level** represented by a given *Slicing Entity*, which divides them into **abstract** and **specific** ones. While all *Deployment-time* and *Runtime Slicing Entities* constitute specific operational objects, this criterion serves as a differentiator of the *Design-time Slicing*

[90] An important remark is that although a given *Design-time Slicing Entity* is referred to in a singular form to denote a logically encapsulated entity, it is frequently composed of multiple parts that cover different aspects of the described entity, e.g. a CSAR archive [163]. Other representations of the *Design-time Slicing Entities* (e.g. in-memory data structures, database tuples, etc.) may be utilized as well.

[91] Depending on the implementation choices, *Slicing Entities* in the *Deployment-time* lifecycle stage do not necessarily need to be represented by dedicated models and objects. They may be represented, for example, by a distinguished operational state of a *Runtime Slicing Entity* that is not instantiated yet.

Entities. In particular, it allows a clear distinction between the notions of **Templates** and **Descriptors**, which in the state-of-the-art are frequently utilized interchangeably, causing confusion as to their meaning. In the elaborated taxonomy, precise semantics of these two terms is defined as follows.

Slicing Entity in the form of a **Descriptor** represents a **detailed, deployment-specific** description of an entity before instantiation. Descriptor precisely defines the entity's internal structure by referencing specific elements out of which it is built, by defining their specific topology, resources assignment as well as by containing their specific configurations, policies and operational behaviors. By containing a full range of information items needed by an Orchestrator, *Descriptors* serve as its operational input. In order to support particular characteristics of the internally utilized elements, as well as the vendor-specific features of the utilized Orchestrator, *Descriptors* are typically technology-specific and non-portable across different Orchestrators, even if based on standardized models.

By contrast, a *Slicing Entity* in the form of a **Template** is a **deployment-agnostic**, **high-level** description of the modeled entity, which leaves all (or the majority) of the deployment-specific aspects **unspecified**. Preferably, Templates stay fully **abstract**, by not referencing any specific elements in the target deployment environments, nor containing any deployment-specific configurations related to such elements. Instead, Templates describe the entity's abstract structure, by only specifying the expected types and characteristics of its internal elements as well as their high-level topology. Not containing any deployment-specific information nor any technology-specific particularities make Templates portable and reusable. According to the understanding provided above, the *Design-time Slicing Entities* in the taxonomy are divided into ***Abstract Templates*** and ***Technology-specific Descriptors***[92], as depicted at level 2 in Figure 20.

Slicing Entities also differ according to the (3) **concretization level of information they contain**, which is another dimension of the presented taxonomy, shown at level 3 in Figure 20. In general, *Slicing Entities* may be divided into ***not fully concretized*** (depicted with thin borders) and ***fully concretized*** (depicted with thick borders). Full concretization means that a given *Slicing Entity* either already is, or can become operational solely with the information it already contains. By contrast, a *not fully concretized Slicing Entity* requires delivery of additional information and potential transformation during the design process to make this possible. The concretization level of *Slicing Entities* utilized in the design process is an important factor from the perspective of its automation. Hence, the characteristics of *Slicing Entities* with a given concretization level, together with the impact of their utilization in the NSI/NSSI design process are described in the following paragraphs.

Slicing Entity in the form of an ***Abstract Template*** describes the modeled entity at a **generic**, high level of abstraction and does not contain any deployment-related information, nor precise, low-level configurations. Thus, during a design process, *Abstract Template* requires **contextualization** against the target deployment environment in order to map its abstractly defined topology of internals to the *Slicing Entities* available in the target deployment environments, to assign their needed resources and to provide their precise configurations. As such, *Abstract Templates* can be considered as patterns for production of *Technology-specific Descriptors* utilized as input to the target Orchestrators. Consequently, utilizing an *Abstract Template* in the design process requires a significant effort before a fully concretized, operationally usable design-time description of the desired entity to be deployed (*Instance Descriptor*, defined in the following paragraph) can be produced based on it.

Slicing Entities in the form of *Technology-specific Descriptors* may come in various levels of concretization that can be classified into ***Semi-concretized Descriptors*** and ***Instance Descriptors***. A *Semi-concretized Descriptor*, typically delivered by vendors along their offered products (e.g. implementation of the AMF MF or a group of RAN MFs organized into a Subnet), is usually **partly concretized**. This means that some or all of its internal elements are explicitly referenced, majority of low-level operational settings are decided and only a limited range of configuration attributes are left with unspecified values to be decided during the design

[92] As an example, even though the OSM project (described in Subsection 2.5.2) utilizes the notion of a Network Slice Template (NST), according to the presented taxonomy it shall be perceived as a *Semi-concretized Descriptor*, as it contains explicit references to concretized elements in the target deployment environments, their detailed configurations, etc. (model can be checked in [101]). On the other hand, the ONAP's NST and NSST, thanks to their abstractly defined internal elements called "Allotted Resources" can be classified as *Abstract Templates* (examples available in [200], [201]).

process (e.g. selection of DF, IL, choice of application configuration and operational profiles, IP addressing schemes, policies, etc.). As such, a particular *Semi-concretized Descriptor* may offer various deployment variants of the described entity and can be reused to create multiple, similarly or identically structured instances[93]. However, the fully concretized **instance-specific configuration must be decided each time an instance shall be created** out of a *Semi-concretized Descriptor*, which is an important factor when considering design process automation. Thus, a *Semi-concretized Descriptor* alone cannot be considered a finalized, operational input for the Orchestrator, as it is not possible to trigger the lifecycle of a concretized instance solely with the information it contains.

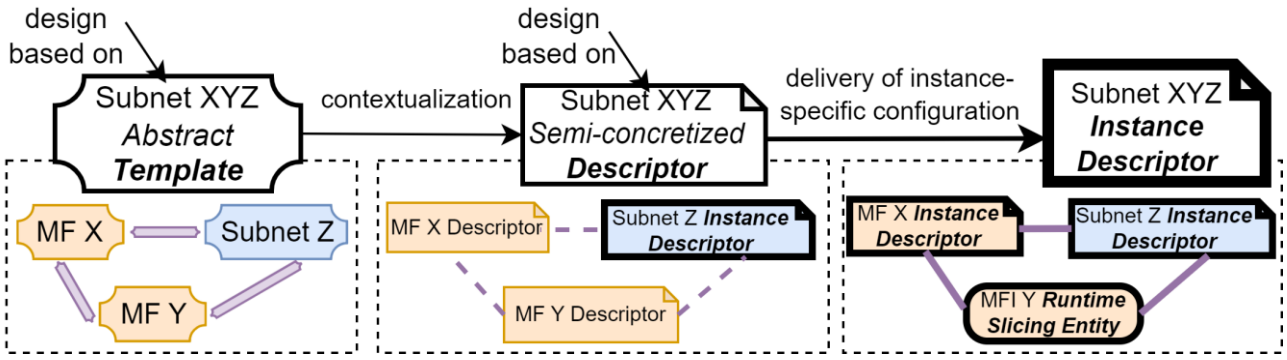


Figure 21. Transitions between *Design-time Slicing Entities* of different concretization levels, which are performed during the design process: from an *Abstract Template*, through a *Semi-concretized Descriptor* until a fully-concretized *Instance Descriptor*.

Such a purpose is fulfilled by a *Slicing Entity* in the form of an ***Instance Descriptor***, which constitutes a logically encapsulated, fully concretized design-time description of an entity that is equipped with a **complete set of instance-specific settings** necessary to trigger the lifecycle of entity's instance in the target Orchestrator. Due to the presence of instance-specific information, *Instance Descriptors* are not reusable, unless all such settings are pruned out of them. But, thanks to their full concretization, they are self-contained, which makes them immediately usable by the target Orchestrators, **without the need of making any additional design-time processing**. As such, these are the *Instance Descriptors* that serve as the finalized, operational input for Orchestrators, based on which triggering the lifecycle of an instance is possible. An *Instance Descriptor* may be created by supplying all the missing, instance-specific settings to a *Semi-concretized Descriptor*, or may be created in a full, dedicated design process based on the abstractly defined structure of an *Abstract Template*, which is schematically presented in Figure 21.

In the case of the *Deployment-time Slicing Entities*, the information they contain is not only *fully concretized*, but also the feasibility of creating an instance with particular configurations has been verified, as the required resources have been pre-provisioned and reserved in the target deployment environments. As such, the concretization level of this information is considered to be higher than in an *Instance Descriptor*. Finally, the already existing, operational *Runtime Slicing Entities* contain information representing their **currently applied configuration** and its **operational** state, which represents the highest level of concretization.

From the perspective of design process automation, the meaning of *Slicing Entity*'s concretization level can be generalized as follows. A *fully concretized Slicing Entity* either already exists in the Orchestrator's runtime inventory (*Runtime* or *Deployment-time Slicing Entity*), or it is possible to commence its lifecycle without conducting any additional design activities (*Instance Descriptor*), because the design decisions determining all the information required for its instantiation and operation have already been made at an earlier moment in time. By contrast, a *not fully concretized Slicing Entity* requires potential transformations during the design process (e.g. contextualization of an *Abstract Template*) and equipping it with fully concretized instance-specific configuration values before an instance can be created based on it. As such, utilization of *not fully concretized Slicing Entities* requires the implementation of more sophisticated processing logic when automating the Network Slice/Subnet design process, which is discussed in the upcoming Chapter 4.

[93] Typically, due to Descriptors' vendor-specifics, they may only be reused within a given technological stack, but are not portable among installations – this characteristic is an important differentiator between Descriptors and Templates, according to the elaborated taxonomy.

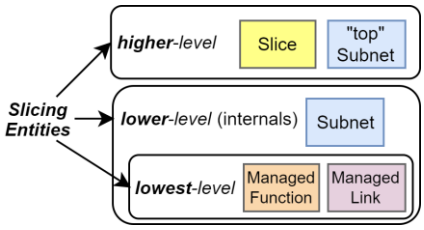


Figure 22. Division of *Slicing Entities* into higher-level, lower-level and lowest-level.

At the fourth level of the taxonomy presented in Figure 20 with different colors, *Slicing Entities* are differentiated according to the ***class*** of entities they represent. Each *Slicing Entity* represents one of four basic classes: an end-to-end Network **Slice** (yellow), a Network Slice **Subnet** (blue), a **Managed Function** (orange) or a **Managed Link** (violet). Collectively, *Slicing Entities* representing Slices and "top" Subnets are referred to in this dissertation as ***higher-level Slicing Entities***. The ones representing MFs, MLs and Subnets other than "top" are referred to as ***lower-level Slicing Entities***, or, alternatively, as Network Slice/Subnet **internals**. Finally, *Slicing Entities* representing MFs and MLs are referred to as the ***lowest-level Slicing Entities***. The classification into *higher-level*, *lower-level* and *lowest-level Slicing Entities* is graphically presented in Figure 22.

A classification criterion that is tightly bound to the *Slicing Entity*'s class is its ***type***, which differentiates *Slicing Entities* of a given class according to their offered set of **characteristics**, like functional or non-functional features, supported QoS attributes, composability with other *Slicing Entities*, etc. This differentiation is schematically depicted at the sixth level of the taxonomy presented in Figure 20. More specifically, Slices perceived as a whole are typed by the SST value, which determines their expected characteristics in terms of offered features, services and offered QoS guarantees[36]. Individual Subnets are typed by the `networkSliceSubnetType` attribute of the `NetworkSliceSubnet` model defined in the 5G NRM, which specifies three coarse-grained values: "Top", "RAN" and "CN". This list may be extended with finer granularity types that represent, for example, a Slice-dedicated subset of RAN/Core MFs, subset of shared RAN/Core MFs (see Figure 8 in Subsection 2.1.6), or any other arrangement. Types of MFs are defined by 3GPP in 5G NRM, by specifying a standardized hierarchy of types[94] inheriting from the `ManagedFunction` defined in the Generic NRM, which has already been extensively covered in Chapter 2. Finally, multiple types of MLs might be distinguished as well, which is defined by the `linkType` attribute of the `Link` model defined in Generic NRM, e.g. `Signalling`, `Bearer`, `OAM&P`[95] or other, custom types.

In the context of design process automation, the class of *Slicing Entities*, together with their type and exposed *Capabilities* (discussed in detail in the upcoming Subsection 3.1.2) serve as important determinants of their **eligibility for utilization in the designed NSI/NSSI**, **enabling their uniform consideration** in the designed NSI/NSSI, regardless of different lifecycle stage they represent, their varying levels of abstraction and concretization. For example, an already running NSSI of type "Shared-RAN", a vendor-delivered *Semi-concretized Descriptor* describing a "Shared-RAN" Subnet or an NSST that abstractly describes a "Shared-RAN" Subnet – all these *Slicing Entities* represent a Subnet of the same type and, as such, all of them can be uniformly considered to be utilized as the "Shared-RAN" part of the currently designed NSI/NSSI. Similarly, a Descriptor of a given SMF function or an already running SMF MFI – both of these *Slicing Entities* represent an MF of type SMF. Hence, both of them can be uniformly considered for utilization in the designed Core NSSI.

The final dimension of the taxonomy, which further emphasizes the heterogeneity of *Slicing Entities* to be embraced while organizing an automated design process, concerns their (5) ***models***, which is schematically presented at the fifth level of Figure 20. As discussed in Section 2.3, only a basic set of management models reflecting the runtime perspective of Network Slicing concepts is provided by the 3GPP's Generic and 5G NRM [32], [80]. Moreover, these models contain only a limited scope of attributes, leaving much leeway for making individual, technology-specific modeling decisions. For these reasons, the models utilized in the particular Network Slicing management solutions, even if grounded in the standardized stubs, are constructed according to the particular technology-specific modeling approaches. Hence, although commonalities may be expected in the narrow set of attributes enforced by the standards, the structure of objects existing in the runtime inventory of an Orchestrator implemented by "Vendor X" will certainly differ from the NSI representation implemented by "Vendor Y" – this is schematically illustrated in Figure 20 by two vendor-specific NSIs deriving from the standardized `NetworkSlice` model. Even less commonalities may be

[94] Besides standardized MF types, custom types representing specialized functionalities (e.g. customized firewalls, DPI modules, etc.) may exist as well.
[95] "OAM&P" is an abbreviation for Operations, Administration, Maintenance and Provisioning.

expected from the design-time models. As discussed in Subsection 2.3.4, currently 3GPP does not cover the design-time aspects, which results in a variety of approaches and models utilized in the Network Slicing management solutions. Nevertheless, some kind of design-time descriptions for entities of all classes must be present in each of such solutions – this is represented in Figure 20 with generic Slice/Subnet/MF/ML Descriptors, exemplified by specific entities performing such a role in the particular solutions.

Based on the inference provided above, *Slicing Entities*' models can be classified into sets denoted in the lower part of Figure 20 by rectangles with red badges: (a) **standardized stubs** of **runtime models**, (b) **technology-specific runtime models**, (c) **technology-specific design-time models**[96] as well as the (d) **abstract, technology-agnostic design-time models**. In a generic case, a Slice/Subnet under design may be composed of *Slicing Entities* belonging to all of the aforementioned sets of models, which is illustrated by the superset (e). Undoubtedly, the heterogeneity of *Slicing Entities*' models (both runtime and design-time), adds greatly to the complexity to be embraced in the Slice/Subnet design process. In the following Subsection, a crucial modeling construct of *Slicing Entities Capabilities* is proposed, which allows mitigating this complexity by enabling uniform treatment of all *Slicing Entities* in the design process despite their high heterogeneity presented in the elaborated taxonomy.

3.1.2. Slicing Entity Capabilities

Despite multiple aspects in which the *Slicing Entities* differ, as detailed above in the elaborated taxonomy, the author recognizes that for the sake of automating the Network Slice/Subnet design process, a common set of characteristics should be exposed by every *Slicing Entity*, regardless of the lifecycle stage it represents, its concretization level, class, type or utilized model. Such characteristics are referred to by the author as the ***Slicing Entity Capabilities*** (abbreviated: *Capabilities*), represented in the taxonomy in Figure 20 by a green rectangle on top of each *Slicing Entity*.

Intuitively, *Capabilities* serve as a kind of a "rating plate" that allows assessment of what to expect from a given *Slicing Entity* if utilized in the designed NSI/NSSI. More specifically, *Capabilities* constitute attributes representing selected characteristics of a *Slicing Entity*[97] that are exposed to allow its **evaluation in terms of meeting the requirements posed against it in the Network Slice/Subnet design process**. From a technical perspective this means that if a particular set of requirements is posed against a *Slicing Entity* in the course of the design process, then a **corresponding** set of *Capabilities* is expected to be exposed by such a *Slicing Entity*, with type and semantics reflecting the posed requirements, as schematically depicted in Figure 23 below. From this moment onwards, a modeling construct representing a set of requirements (e.g. the `ServiceProfile`, `SliceProfile` standardized data structures or any other, customized structures containing requirements) will be referred to as ***Requirements***, and a single requirement contained in such set will be referred to as a ***Requirement attribute***. Correspondingly, *Slicing Entities* expose ***Capabilities*** constituting sets of ***Capability attributes***.

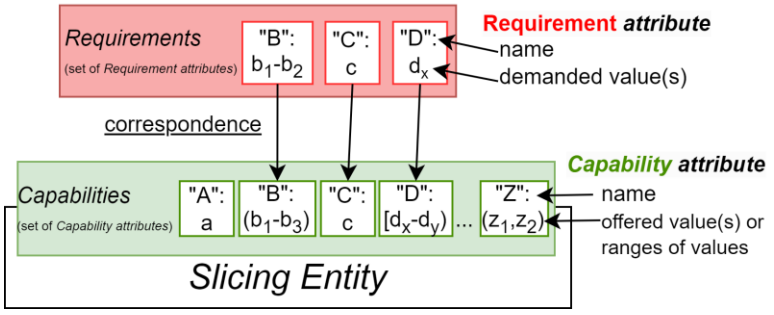


Figure 23. Correspondence between *Requirements attributes* and *Capabilities attributes* in terms of name, type and semantics.

It shall be underlined that not all the detailed characteristics of a *Slicing Entity* must be exposed as *Capabilities*, but only those that allow the assessment whether it meets the *Requirements* posed against it. For example, if an Orchestrator is meant to support the `dLLatency` SLS-satisfaction *Requirement* in the input `SliceProfile`[98],

[96] The design-time models encompass the runtime models, in order to allow the inclusion of sharable *Runtime Slicing Entities* in the designed NSI/NSSI.

[97] Although in this Chapter the concept of *Capabilities* is presented purely in the context of *Slicing Entities*, other entities may expose *Capabilities* too, e.g. the deployment environments considered in the design process, which is discussed in Subsection 4.3.5.

[98] Here it shall be reminded that supporting each single SLS attribute may require complex, long-lasting development at various levels of the orchestration stack, which has been discussed in Subsection 2.3.2.

then all the utilized *Slicing Entities* that represent Subnet should expose a corresponding `dLLatency` ***SLS-satisfaction Capability***, stating the **achievable value** of this characteristic.

Here it needs to be pointed out that not only the high-level SLS-satisfaction *Requirements* contained in the input `ServiceProfile`/`SliceProfile` are posed against the *Slicing Entities* during the design process. As it will be described in the upcoming Subsection 4.3.5, when the designed NSI/NSSI is being assembled from the *lower-level Slicing Entities* (Subnets, MFs, MLs), there are *Requirements* of lower-level semantics stated against them. This includes lower-level *SLS-satisfaction Requirements*, but also *functional* and/or *composability Requirements*, which allow ensuring that the *lower-level Slicing Entities* will fit each other and compose a well-functioning whole. As such, depending on the *Requirements attributes* to be supported during the design process, the corresponding *Capabilities attributes* to be exposed by the particular *Slicing Entities* may be of various semantics, representing both high and low-level characteristics.

Taking the aforementioned aspects into account, three general groups of *Capabilities* exposed by *Slicing Entities* can be distinguished: (a) ***functional*** *Capabilities*, (b) ***composability*** *Capabilities* and, most importantly, (c) ***SLS-satisfaction*** *Capabilities*. More specifically, the (a) *functional Capabilities* indicate what behavior does a given *Slicing Entity* deliver, the (b) *composability Capabilities* allow for verification whether it will fit the topology of elements in the designed NSI/NSSI, and the (c) *SLS-satisfaction Capabilities* indicate how well it performs (in the case of *Runtime Slicing Entities*), or declares to perform (*Design-time* and *Deployment-time Slicing Entities*). The mentioned groups of *Capabilities* and the interpretation of their values depending on the *Slicing Entity*'s lifecycle stage and concretization level is further discussed in the upcoming Subsection 3.1.4.

Introducing the modeling construct of *Capabilities* with the semantics described above is essential from the perspective of design process automation, as it allows for **conveying an important piece of domain and operational knowledge that is needed for hands-free processing of *Slicing Entities*** in the basic design activities (i.e.: *evaluation*, *selection* and *Requirements decomposition*, which are detailed in the upcoming Section 4.3). Hence, determining a specific set of *Capabilities* attributes to be exposed by the *Slicing Entities* utilized in a given Orchestrator constitutes an important step in their preparation for hands-free processing in automated design activities. The duty of determining the concretized sets of *Capabilities* attributes, including credible determination of their offered values, lies with different parties depending on the *Slicing Entities'* lifecycle stage and concretization level - this aspect is further discussed in the upcoming Subsection 3.1.3 and 3.1.4.

Although the exposition of *Capabilities* by *Slicing Entities* may seem like a natural need to allow designing NSIs/NSSIs capable of satisfying the SLS *Requirements* passed as input to the design process, the author's analysis of the currently standardized Network Slicing management models (comprehensively presented in Section III.G.2 of [30]) reveals that currently these models do not contain any explicit notion of *Capabilities*. The author finds it surprising, because due to the SLS *Requirements*-centricity of the `allocateNsi`/`allocateNssi` operations, **exposition of *Capabilities* is among the most critical factors** for the organization of an SLS *Requirements*-oriented design process and its automation. First of all, if a particular *Slicing Entity* does not expose its *Capabilities*, then it is unknown what can be expected from it when utilized in the designed NSI/NSSI which, in turn, makes it unknown what level of SLS *Requirements'* satisfaction can be achieved. Secondly, the modeling construct of *Capabilities* constitutes an abstraction layer that enables uniform consideration of heterogenous *Slicing Entities* during the design activities, despite significant differences discussed in the presented taxonomy. Finally, *Capabilities* enable perceiving *Slicing Entities* as **encapsulated individuals** that are viewed only from the perspective of characteristics essential in the design process. This allows for concealing their internal complexity, relieving the party considering their utilization in the designed NSI/NSSI from possessing extensive knowledge about their low-level details. For the enumerated reasons, the author postulates in [30] that *Capabilities* with the proposed semantics and uniform representation **should be considered for standardization** to be **inherently present** in all of the *Slicing Entities'* models, both runtime and design-time.

While the exposition of *Capabilities* is among the most important enablers of NSI/NSSI design process automation, there are other pieces of domain and operational knowledge that need to be modeled and

captured to achieve automated processing in other essential design activities. Appropriate modeling constructs that enable depositing such knowledge, structured to allow its leveraging in an automated logic, are proposed in the upcoming Sections of this dissertation.

3.1.3. Slicing Entity Configuration Profiles

When discussing *Capabilities*, it is important to note that a single *Slicing Entity* may exhibit different values of the offered *Capabilities*' attributes, depending on its current or possible to be applied **configurations**. This is because deciding even on a single, influential configuration setting may radically change the *functional*, *composability* or, most importantly, the *SLS-satisfaction Capabilities* of a particular *Slicing Entity*. Consequently, a change in configuration may influence the *Slicing Entity*'s potential for fulfillment of the input SLS *Requirements*, which is arguably the most important factor when designing and, later on, operating an NSI/NSSI. For example, by configuring a *Semi-concretized Descriptor* of a given SMF Function with a limited amount of memory and computing resources, it is only capable of managing a reduced count of simultaneous PDU sessions (e.g. up to 100 000) and cannot offer the full extent of its expected functionalities, but only their subset. However, with the increased amount of memory and assignment of more computing cores, it can manage a higher count of simultaneous PDU sessions (e.g. up to 1 000 000) and offer full extent of functionalities.

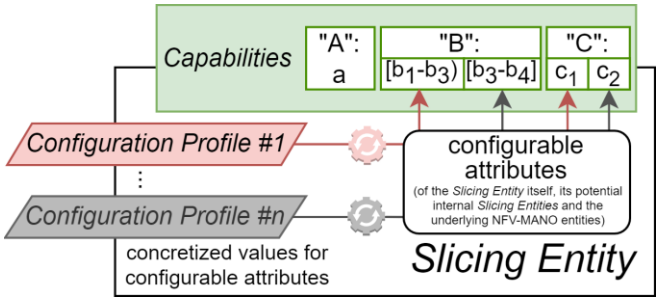


Figure 24. *Confiuration Profiles* containing sets of concretized values for the configurable attributes of a *Slicing Entity* and its internals.

A set of concretized settings for the configurable attributes of a given *Slicing Entity* that results in the exposition of *Capabilities* with particular values is referred to by the author as its ***Slicing Entity Configuration Profile*** (abbreviated: *Configuration Profile*), which is schematically presented in Figure 24. The concept of a *Configuration Profile* can be perceived similarly to the ETSI-standardized `NsProfile`, `VnfProfile` or `VirtualLinkProfile` data structures [84], but extended towards covering the **application-layer configurations** and supplemented with the concept of *Capabilities*, which are not covered by the ETSI NFV-MANO constructs. This means that besides containing configuration values that may influence the performance or functionality of the NFV-MANO entities underlying a particular *Slicing Entity* [99], a *Configuration Profile* should also contain **values for the application-layer settings that may impact the *Capabilities*' values**[100]. Due to a potentially large spectrum of such influential configuration settings, it is advisable that the Vendor of a given *Slicing Entity* (or, alternatively, the Operator that plans to utilize it) predefines a limited number of *Configuration Profiles* - each specifying a coherent, sensible set of settings, utilization of which makes the *Slicing Entity* **exhibit concretized values of the offered *Capabilities***.

For example, a given *Semi-concretized Descriptor* of a RAN Subnet delivered by a given Vendor may contain two predefined *Configuration Profiles*: with eMBB-optimized or mIoT-optimized settings. Besides determining the configuration values for the underlying NFV-MANO entities, each *Configuration Profile* dictates particular values of the application-layer configurations for its internal MLs (e.g. routing protocol settings optimized for high throughput or low-latency) and MFs (e.g. scheduling weights applied by gNB-DU, percentage of PRBs[8] assigned in cells for a particular SST[36] value, etc.). Applying a given *Configuration Profile* makes the considered *Semi-concretized Descriptor* exhibit particular values of the offered *Capabilities*, e.g. session handling functionalities (*functional Capabilities*), exposition of the `EP_N2` and `EP_N3` endpoints (*composability Capabilities*) and concretized values of the achievable overall throughput in Slice, throughput per UE, latency and others (*SLS-satisfaction Capabilities*).

A given *Configuration Profile* may also include values for attributes that do not necessarily impact the *Capabilities'* values, but relate to the particular way of configuring a given *Slicing Entity*, e.g. a specific

[99] For example, these are the choices of a Deployment Flavor (DF), Instantiation Level (IL) [84], [81], etc. According to the classifications performed in the upcoming Subsection 3.3.2, these are the values for attributes related to the "platform" and "virtualized entity" layers (see column E in Table 5).

[100] According to the classifications performed in the upcoming Subsection 3.3.2, these are values for "impactful" attributes (see column G in Table 5).

frequency of performance measurements collection adjusted to the eMBB or mIoT *Configuration Profile*, a specific scheme of alarm thresholds or hysteresis, etc. The larger spectrum of configuration attributes is covered with concretized values in the *Configuration Profiles*, the less effort is needed during the design process to *configure* a given *Slicing Entity* for instantiation and operation.

The proposed concept of *Configuration Profiles* is another modeling construct that allows for **depositing the Operator's or Vendor's expert domain and operational knowledge**, with the aim of leveraging it for automating the particular activities in the design process. More specifically, *Configuration Profiles* play an important role in the automation of the *evaluation* activity and, most importantly, the *configuration* activity, which are detailed in the upcoming Section 4.3. Credibility of the *Capabilities* values offered by the given *Configuration Profiles* should be verified by operational tests of the considered *Slicing Entity* with particular configuration values applied, which is further discussed in the following Subsection 3.1.4.

As a final remark, it shall be noted that the availability of thoughtfully determined *Configuration Profiles* with credible *Capabilities*' values is not only vital for automating the design process, but may also be utilized for automating the operational lifecycle of *Slicing Entities*. In particular, a given *Slicing Entity* may offer several *Configuration Profiles* with a gradually increasing "amount" of consumed resources and specific application-layer settings, resulting in the offering of gradually increasing *SLS-satisfaction Capabilities*' values. Transition to another *Configuration Profiles* can be triggered by the SLS-assurance policies and closed-loop mechanisms of the utilized Orchestrator[101]. For example, a *Semi-concretized Descriptor* of an SMF function may offer *Configuration Profiles* named "min", "mid" and "max" that offer support for a gradually increasing count of simultaneous PDU sessions, depending on the CPU, memory and application-layer settings. The "min" Configuration Profile can be utilized at the beginning of the *Slicing Entity*'s lifecycle, when it handles a limited load incurred by the SLS *Requirements* assigned to it. But, when the load increases, the Orchestrator may switch the *Configuration Profile* to "mid" or "max" in order to dynamically adjust the ratio of consumed resources to the increased operational demands and the conditions experienced in the operated telecommunication network.

3.1.4. Interpretation of Slicing Entity Capabilities' values

When implementing the logic of an automated design process, the **interpretation** of *Capabilities*' **values** offered by the *Configuration Profiles* of a given *Slicing Entity* needs to be taken into account. It is because their semantics **differs depending on the lifecycle stage** a given *Slicing Entity* represents and its **concretization level**. In general, the *Capabilities*' values exposed by the *Configuration Profiles* of *Design-time* or *Deployment-time Slicing Entities* represent **declarations**, whereas in the case of *Configuration Profiles* of *Runtime Slicing Entities* these are their **currently offered** *Capabilities*, indicated by their operational state.

More specifically, the values of the (1) *functional* and (2) *composability Capabilities* exposed by *not fully concretized Slicing Entities* represent the declarations of all the functional features that are possible to be enabled and all the endpoints a given *Slicing Entity* may expose, if such *Slicing Entity* is utilized operationally with a particular *Configuration Profile* applied. In the case of *Instance Descriptors* and *Deployment-time Slicing Entities*, these are still declarations, but representing the already configured, selected deployment variants of the offered functionalities and the endpoints configured for operation. Finally, in the case of the *Runtime Slicing Entities*, these characteristics represent the actual, currently offered functionalities, and currently exposed endpoints (exposition of which may change over its lifecycle, e.g. as a result of reconfiguration).

When considering the (3) *SLS-satisfaction Capabilities*, the values exposed by the currently exhibited *Configuration Profile* of a *Runtime Slicing Entity* indicate the real, monitored state of SLS *Requirements* satisfaction (e.g. their corresponding KPI metrics, potentially supplemented with historical or predicted values). *Capabilities*' values of its other, not currently exhibited *Configuration Profiles* are declarations of the achievable metrics as contained in its underlying *Instance Descriptor*. *Configuration Profiles* of a *Design-time*

[101] The configuration settings contained in a particular *Configuration Profile* may switch on the *Slicing Entity*'s internal SLS-assurance features, if available. This may entail its autonomous adaptation to the experienced conditions (e.g. autoscaling, autohealing, etc.), taking into account the KPIs or measurements a given *Slicing Entity* monitors by itself. In such a case, the *Capabilities'* values offered by a *Configuration Profile* may be specified as ranges.

or a *Deployment-time Slicing Entity*, on the other hand, can only define declared values of the *SLS-satisfaction Capabilities*, achievable if an instance is created out of them. As such, *Abstract Templates*, due to their abstractness and lack of concretized configurations, may only specify a "dummy" *Configuration Profile* (not containing any configuration values), which states estimated ranges of the *SLS-satisfaction Capabilities* as well as the expected functionalities and endpoints of a *Slicing Entity* which will be created by contextualizing such an *Abstract Template*. In contrast, values declared by the *Configuration Profiles* defined in *Semi-concretized Descriptors* shall be grounded in a tested, credibly assessed performance of the described entity if it gets instantiated from such *Semi-concretized Descriptor* utilizing a particular *Configuration Profile*.

As an illustrative analogy, the *SLS-satisfaction Capabilities*' values exposed by a particular *Slicing Entity Configuration Profile* may be compared to the specification of a computer game which declares the delivery of particular QoE measures (e.g. level of graphics details, achievable rate of frames-per-second, etc.) if the game runs in particular environmental conditions (e.g. amount of memory, operating system, class and extensions supported by the graphics card, utilized drivers and their configuration, etc.) and with its specific configuration (e.g. enforced level of graphics details, etc.). Similarly, in the Network Slicing domain, the Customer's SLS *Requirements* in a particular scenario may state, for example, a strict *Requirement* on the latency of the user-plane communication. Such *Requirement* may not be satisfied by utilizing an existing purely software-based UPF MFI (*Runtime Slicing Entity*), which advertises its poor runtime KPI values. However, there might be a *Semi-concretized Descriptor* of another UPF, offering a *Configuration Profile* which takes advantage of hardware acceleration (e.g. *Single Root I/O Virtualization* (SRIOV) [155]) and declares *Capabilities*' values (e.g. promised KPIs values) which should allow to meet the posed SLS *Requirements*, if a UPF instance is created with the configuration settings specified in such *Configuration Profile*.

Table 4. Semantics of *Capabilities*' values exposed by the *Configuration Profiles* of *Slicing Entities* with a given concretization level.

Slicing Entity **concretization level**	contained ***Configuration Profiles***	**interpretation** of *Capabilities*' **values** of *Configuration Profiles*	**semantics** of *Capabilities*' **values**		
			functional	*composability*	*SLS-satisfaction*
Abstract Template	single "dummy" *Configuration Profile*	**wide ranges** of values (values in the range **expected** to be delivered when contextualizing this template)	**declarations** of the **expected** functionalities	**declarations** of endpoints **expected** to be supported	**ranges** of values for the **expected** level of SLS satisfaction
Semi-concretized Descriptor	**several** *Configuration Profiles* specifying sensible, coherent set of configuration values	**narrowed ranges** of values **possible to be exposed** (preferably - experimentally verified)	**declarations** for the totality of **available** functionalities **possible to be enabled**	**declarations** for the totality of endpoints **possible to be supported**	**declared**, **achievable ranges** of values for SLS satisfaction (credibly assessed by operational testing)
Instance Descriptor	**single** *Configuration Profile* **configured**, transition to other *Profiles* possible when reconfigured	**concretized** values **to be exposed** (preferably - experimentally verified)	**declarations** of functionalities **configured** to be enabled (and possible to be enabled via reconfiguration)	**declarations** of endpoints **configured to be supported**	**declared, concretized** level of SLS satisfaction
Deployment-time Slicing Entity	**single** *Configuration Profile* **configured**, transition to other *Profiles* possible when reconfigured	**concretized** values **to be exposed** (**authenticated** by **reservation** of the necessary resources)	**declarations** of functionalities **pre-provisioned** to be enabled (and possible to be enabled via reconfiguration)	**declarations** of endpoints **pre-provisioned** and **reserved**	**declared, concretized** level of SLS satisfaction (**authenticated** by **reservation** of the necessary resources)
Runtime Slicing Entity	**single** *Configuration Profile* **currently utilized operationally** transition to other *Profiles* possible when reconfigured	**real**, **monitored** values (exhibited with the currently applied *Profile*, in the current state of the deployment environment)	**currently enabled** functionalities (and possible to be enabled via reconfiguration)	**currently operational** endpoints	**currently monitored KPI values** and **PM/FM metrics**, measuring SLS satisfaction (supplemented with historical, predicted trends)

The interpretation and semantics of the *Capabilities*' values exposed by the *Configuration Profiles* of a *Slicing Entity* depending on its concretization level is summarized in Table 4. This interpretation needs to be taken into account when implementing and configuring an automated design solution, as utilization of *Slicing Entities* with a particular concretization level may be preferrable in certain circumstances.

3.2. Definitions of fundamental Network Slicing design-time aspects

The previous Section discussed how the proposed concept of a *Slicing Entity* and its related notions of *Capabilities* and *Configuration Profiles* allow for systematic classification of the building blocks out of which an NSI/NSSI may be designed. In this Section, in turn, it is shown how the introduction of these concepts

enables formulating neat definitions of two fundamental design-time aspects that have not been formally specified in the state-of-the-art so far. More specifically, in the following Subsections, the author proposes generic **definitions** of the (1) **Network Slice/Subnet design process** as well as its (2) **expected outcomes**. Formulation of these missing definitions serves as an important signpost in the upcoming analyzes and further elaboration of concepts needed for the construction of an automated design process.

It is worth noting that due to the 1-to-1 relationship between the Slice and Subnet models at the technical level (discussed in Subsection 2.3.3) and the recursive nature of a Subnet, the provided definitions apply equally to the end-to-end Slices as well as to the Subnets, regardless of their nesting level.

3.2.1. Definition of the Network Slice/Subnet design process

The **Network Slice/Subnet design process** is defined by the author as *a thoughtful analysis of the input SLS requirements and their potential decomposition, based on which an informed evaluation, selection and determination of configuration of Slicing Entities available in the target deployment environments is performed*, *in order to assure that the NSI/NSSI decided to be newly constructed out of them or the NSI/NSSI selected to be reused will, by dedicating a determined share of Capabilities of its Slicing Entities, (1) maximize the chances of satisfying the Customer's input SLS requirements, and (2) optimize its structure and operation according to the Operator's operational and Provider's business objectives*.

While the (1) first aspect of the provided definition represents the expectations of the Customer who orders or triggers NSI/NSSI allocation and expects preservation of the SLS contract stated in the input `ServiceProfile`/`SliceProfile`, the (2) second aspect takes into account the Operator's and Provider's perspectives, who may wish, for example, to optimize the economic measures like profit, OPEX or CAPEX, by dedicating the minimum share of *Slicing Entities*' *Capabilities* and their underlying resources in the designed NSI/NSSI that will be sufficient for satisfying the Customer's SLS *Requirement*s. As such, the level of applied optimizations, or lack of them, depends on predefined objectives.

A thorough elaboration of the design process definition is provided in Chapter 4. Before that, it needs to be clearly stated what are the expected Slice/Subnet design outcomes, which is defined in the following Subsection.

3.2.2. Definition of the Network Slice/Subnet design outcomes

Once the design process is accomplished, its outcomes should constitute an immediately usable operational description of an NSI/NSSI that has been decided to be newly provisioned or a reference to an NSI/NSSI to be reused, as well as a specification of the share of *Capabilities* of the utilized *Slicing Entities* to be devoted for satisfying the Customer's input SLS *Requirement*s. Importantly, in order to support a hands-free, fully automated allocation process, the design outcomes need to be ***fully concretized***, not requiring performing any further design activities to commence the lifecycle of an appropriately configured NSI/NSSI and its internal constituents.

To serve such a purpose, the **outcomes of the Network Slice/Subnet design process** are defined by the author as *a set of interdependent Instance Descriptors specifying the NSI/NSSI to be newly provisioned or the identification of the NSI/NSSI to be reused, supplemented with the specification of the determined share of Capabilities of the utilized Slicing Entities to be devoted to satisfy the Customer's input SLS requirements, based on which the utilized Orchestrator is able to: (1) instantiate and/or configure the NSI/NSSI, its internal Slicing Entities and their underlying entities in the NFV-MANO layer, and (2) operate the NSI/NSSI and its internals, maximizing the chances of Customer's SLS requirements' satisfaction by exploiting the determined share of Slicing Entities' Capabilities*.

It must be stressed that neither the design process, which aims at determining an NSI/NSSI satisfying the input SLS *Requirement*s, nor the design outcomes expressing the design decisions **can guarantee** such satisfaction. It is because the ultimate goal of the design outcomes is essentially to convey the decisions made during the design process, which, depending on the Operator's and Provider's policies, may prioritize different objectives. For example, if during the design process it is decided to dedicate a higher or lower share of *Slicing Entities' Capabilities* (e.g. because of customer's priority, to minimize resources utilization, cost

and/or to increase profit), the Orchestrator shall operate the NSI/NSSI to strive for SLS satisfaction, but within the determined borders of *Slicing Entities Capabilities'* utilization. Thus, it is the role of the Orchestrator to strive for preserving the SLS guarantees according to the design decisions conveyed in the design outcomes.

3.3. Modeling of Slice and Subnet *Instance Descriptors*

According to the definition provided in the previous Subsection, when the design decision is to provision a new NSI/NSSI, the design process should produce a set of *Instance Descriptors* conveying a complete set of information required by the utilized Orchestrator to instantiate and operate all the entities comprising it, both at the 3GPP Network Slicing management layer and in the underlying ETSI NFV-MANO layer. However, while the design-time description of the underlying NFV-MANO entities are well-defined, **there is currently no standardized specification of how a description of a Slice or Subnet before instantiation should look like**, as discussed in Subsection 2.3.4. To this end, this Section is devoted to the analysis of the information items to be contained in such a description in a generic case and the proposition of its standards-compliant structure that leverages the already proposed modeling constructs of *Slicing Entities*, *Capabilities* and *Configuration Profiles*.

It must be pointed out, though, that the author's intention is **not** to dictate the exact, detailed specification of the missing design-time models with a precision down to the level of attributes' exact names, data types, cardinalities, relationships etc. Although such specification is highly needed, the author believes that it needs to be established by a broad standardization consensus, which is unfortunately not being pursued at present. Instead, in support of such endeavors, the author contributes a generic, high-level structure of the missing Slice/Subnet *Instance Descriptors*, proposed based on the inference provided in this Section. More specifically, in Subsections 3.3.2, 3.3.3 and 3.3.4, the **information items** that shall be present in the Slice/Subnet *Instance Descriptors* are **identified** and **classified** into **equivalence classes** according to several criteria important in the context of design process automation. Then, in Subsection 3.3.5, a **layered distribution** of the identified information items is discussed, based on which a **high-level structure** for the missing 3GPP-ETSI standards-compliant Slice and Subnet *Instance Descriptors* is proposed. Before these contributions are presented, several important remarks regarding the role of a Slice/Subnet *Instance Descriptor* are firstly discussed in the following Subsection 3.3.1.

As a final disclaimer, the author informs that the described study, in its broader extent, is presented in [30] and its supplementary material – the following Subsections present its abbreviation version and main conclusions.

3.3.1. Role of a Slice/Subnet Instance Descriptor

The role of any entity's Descriptor arises directly from the role of the described entity in its operational domain. Hence, to clarify the role of a Slice/Subnet *Instance Descriptor* modeled in the current Section, it shall be well understood what is the primary operational role of the NSI/NSSI objects in the managed telco network.

To comprehend it properly, it needs to be realized that a given NSI/NSSI is essentially a **composite** management entity that by itself does not deliver any functional or non-functional features - all of them are delivered entirely by the particularly configured MFs and MLs they are composed of. As such, a given NSI/NSSI object is a purely virtual entity that barely **puts them together** and **supplies their configurations** in order to assure their expected functioning and SLS fulfillment of the whole group. Thus, the role of the NSI/NSSI object is to **group**, **hierarchize** and **encapsulate** the contained *lower-level Slicing Entities* in order **to allow convenient management and supervision of such a group according to the stated SLS requirements**.

Based on this reasoning, the role of a Slice/Subnet *Instance Descriptor* can be induced, which is essentially to **convey** all the **concretized design decisions** related to the configuration of the encapsulated *Slicing Entities* perceived both individually and as a coordinated group. Thus, to fulfill the specified role and meet the statements of the design outcomes definition, the author identifies that a Slice/Subnet *Instance Descriptor* needs to convey information items pertaining to the coarse-grained aspects defined below.

Aspect 1: NSI/NSSI as a set of individual internals:

1.1. identification of internal *Slicing Entities*, their topology and decided geolocation in PoPs;
1.2. decided configuration values concretizing all internal *Slicing Entities*.

Aspect 2: NSI/NSSI perceived as a whole:

2.1. determined Slice/Subnet-level *Configuration Profiles*, aggregated *Capabilities* attributes and their concretized values depending on *Configuration Profiles*;
2.2. Slice/Subnet-level SLS-assurance configurations.

Determination of information items addressing these aspects is provided in the following two Subsections.

3.3.2. Determination of information items pertaining to the internals of the designed NSI/NSSI

To address aspect 1.1, a Slice/Subnet *Instance Descriptor* needs to precisely **identify** the *Slicing Entities* to be utilized as its **internals**. Deciding on the identity of *Slicing Entities* to be utilized in the designed NSI/NSSI (performed in the *selection* activity described in the upcoming Subsection 4.3.3) not only determines the structure and features of the designed NSI/NSSI, but also entails its **footprint,** as each of the pointed *Slicing Entities* with a given configuration consumes a certain fraction of resources in the underlying virtualization platform. Moreover, it is crucial to convey information about the target **geolocation** of *Slicing Entities* representing MFs or nested Subnets, as well as the **topology** of MLs between their endpoints. The geolocation information is needed not only to allow the Orchestrator to transmit the *Slicing Entities'* software artifacts and onboard them in the target PoPs, but also to assure the desired geographical coverage of the designed NSI/NSSI by appropriate distribution and configuration of the gNB MFs. Thoughtful determination of MFs/Subnets geolocation is also important when optimizing the performance of the designed NSI/NSSI[102]. Lastly, the topology of MLs, combined with the decided geolocation of MFs and Subnets to be connected, determines which of the underlying Virtual Links (VL) in the NFV-MANO layer shall be established as intra-DC, inside of a particular PoP, or between PoPs in the form of Data Center Interconnection (DCI)[103] [156], [157].

The conducted study points out to the fact that the internal *Slicing Entities* decided to be utilized in the designed NSI/NSSI can be in various lifecycle stages: *Design-time*, *Deployment-time* or *Runtime*. As such, they can be of different concretization levels. To enable their hands-free instantiation, configuration and operation, their full concretization must be assured. Thus, to address aspect 1.2, a finalized Slice/Subnet *Instance Descriptor* **needs to store instance-specific settings for all of the internally utilized *not fully concretized Slicing Entities***. The study further reveals that the particular set of instance-specific configuration values to be stored depends on their class, type and concretization level.

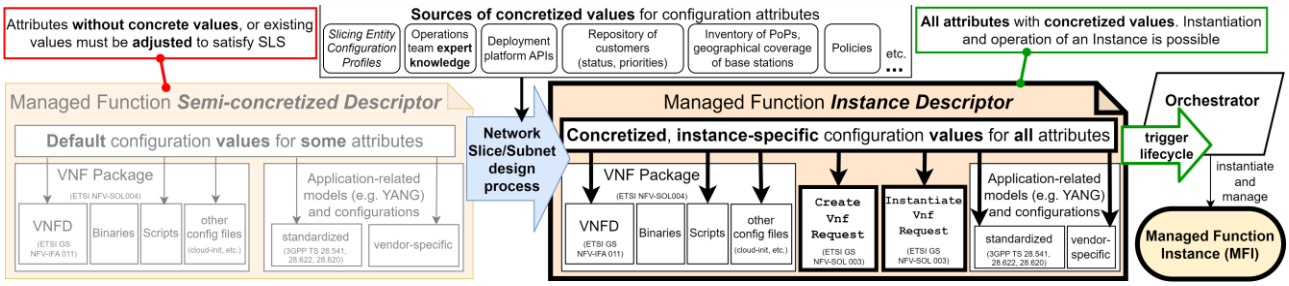


Figure 25. Example of an MF *Semi-concretized Descriptor* concretized during a design process with configuration values for attributes required by its encapsulated entities and data structures. The effectively produced MF *fully concretized Instance Descriptor* allows unattended instantiation and operation of an MFI.

For example, as schematically depicted in Figure 25, a complete set of configuration attributes required for the instantiation and operation of a given MFI out of its *Semi-concretized Descriptor* includes its application-

[102] Geolocation of MFs is especially important for NSIs/NSSIs of types URLLC, V2X or HMTC, or in the Multi-access Edge Computing (MEC) use cases [74], [75]. Placement of particular MFs close to the end-user UEs (e.g. UPF in the RAN PoPs) is crucial to increase the QoE in the latency-sensitive applications (gaming, V2X, remote surgeries, etc.) or to assure immediate content dissemination, e.g. Content Distribution Networks (CDN), live streaming, etc.

[103] Establishment of a sliced DCI may be the duty of a specialized controller, e.g. the IETF Network Slice Controller (NSC) [181].

layer settings (i.e. values for the 3GPP-standardized application attributes[104] conveyed e.g. in YANG models [91], as well as for any vendor-specific extensions), but also all the configuration values essential to instantiate and operate its underlying VNF(s) in the NFV-MANO layer[105]. All of the attributes of the aforementioned models need to be supplied with concretized values – otherwise automated instantiation and operation of an MFI is not possible. To determine appropriate values for all required attributes, a variety of sources may need to be accessed and examined during the design process, as schematically depicted in Figure 25.

Similarly, a *not fully concretized Slicing Entity* representing a Subnet shall not only be supplied with the Subnet-level configurations (discussed in the upcoming paragraphs), but also with concretized configuration values for its `Network Service` (NS) that underlies it in the NFV-MANO layer[106]. Finally, the same applies to the *not fully concretized Slicing Entities* representing MLs, which are typically realized on top of the VL abstraction in the NFV-MANO layer. As an example, if an MLI shall span different PoPs, the underlying VL may require the establishment of a DCI, which can realized by the ETSI-defined `MSCS` [82], [83]. Thus, in order to enable the Orchestrator to establish the full networking stack for an MLI, a complete set of configuration values required by its underlying `Link` model [80] as well as values for the design-time descriptions of the underlying NFV-MANO entities[107] must be present in the design outcome as well.

In order to determine information items that could serve the purpose of providing concretized configurations for the internal *Slicing Entities*, the author performed a comprehensive study of attributes present in the currently standardized management models of all *Slicing Entities* classes, as well as in the corresponding modeling constructs of their underlying NFV-MANO entities. A general contribution of the study is the **augmentation** of the 3GPP Network Slicing standards with the **design-time perspective of the management models**, which is currently left unspecified in these standards. Among the main results, Table 5 presents a broad classification of configurable attributes contained in these models, from several perspectives that are important for organizing an automated design process. In particular, for each of the studied attribute, it has been determined:

A. if it has a **default value**, or its value must be decided and delivered during the design process;
B. if the already specified value (default, or previously set) **can** or **shall be modified** during the design process in order to meet the design objectives, e.g. to assure SLS *Requirement*s fulfillment, to optimize resources utilization, reduce cost, etc.;
C. what are the possible **sources of value** for such attribute;
D. how the **automated determination** and **delivery of value** for a given attribute may be attempted.

Additionally, for each studied attribute, the **level of difficulties** that may need to be faced **when determining its value** has been assessed as follows:

- **no effort** or **trivial** (represented with green color of cells in Table 5) – this means, for example, that a default value exists and should not be modified, etc.
- **average** (yellow) – this means, for example, a potential need of modifying the default value, a need to fetch values from external systems or perform advanced calculations, etc.
- **significant** (red) – this means, for example, that no default value exists, value is strictly dependent on the input SLS *Requirement*s and objectives of the design process, it is determined by combining information from multiple sources, by complex processing, requires human intervention or expert knowledge, etc.

[104] These are the attributes present in the inheritance hierarchy of Information Models: `Function_` IOC defined in [202] `ManagedFunction` IOC defined in Generic NRM [80] and type-specific models `XyzFunction` (e.g. `AMFFunction`, `GNBCUCPFunction`, etc.) defined in 5G NRM [32].

[105] These are the configurable attributes specified in the VNFD [81], any customized scripts or system-level settings contained in the VNF Package [86], as well as the `CreateVnfRequest` and `InstantiateVnfRequest` data structures [88], required to trigger the lifecycle of such VNF(s).

[106] This includes the configurable attributes in the corresponding NSD [84] and the referenced VNFDs, as well as the configuration values needed for the creation/instantiation of an NS instance, conveyed inside by `CreateNsRequest`/`InstantiateNsRequest` [89] data structures, respectively.

[107] i.e. `VirtualLinkProfile` [84] and, potentially, `MscsProfile`/`MscsData` [82] data structures.

Table 5. Classification of config attributes that need to be supplied with concretized values in *Slicing Entities* of a given class to be *fully concretized*.

Slicing Entity Class	**Model of the underlying entity** or **data structure** which **requires configuration**		**Configuration attribute(s)** in the model of the underlying entity or data structure (descriptions of attributes' semantics to be found in the referenced specifications)	**A** Does/should the attribute have a **default value**?	**B** Should/can the attribute value be **provided** or **edited** during the design process?	**C** **Source of value** during design process (if value not already present in the underlying entity or can/shall be modified)	**D** Possibilities for **automated value determination** and/or **delivery** (if value can/shall be specified/edited during the design process)	**E** **Layer** to which the attribute applies: platform	virtualized entity	application	**F** **Dependency** on configuration values of other *Slicing Entities*: independent	dependent	**G** **Impact** on *Slicing Entity Capabilities*' values: impactful	not impactful	**H** **Slice-relevancy**: slice-agnostic	slice-influenced	slice-specific
Managed Function (MF)	**VNF**-level attributes	**VNFD** [84]	`Virtual Compute Desc`	`virtualCpu virtualMemory`	yes	no	value already present (defined by the Vendor of the MF and its underlying VNFD)	no need	✓			✓		✓		✓		
			`VnfExtCpd`	`cpProtocol.addressData`	possibly (if *ipAddressAssignment Subtype=DYNAMIC*)	possibly	IP addressing service of the underlying deployment platform or Operator	provided by a pre-configured IP-addressing service of the virtualization platform or from a predefined MF-level *Slicing Entity Configuration Profile*		✓		✓			✓	✓		
			`Vnf Virtual LinkDesc`	`virtualLinkDescFlavour.qos (latency, packetDelayVariation, packetLossRatio)`	yes, should be	possibly	Vendor of the MF and its underlying VNFD, or Operator	provided from one of the predefined MF-level *Slicing Entity Configuration Profile* (*Capabilities* matched against the incoming, decomposed SLS *Requirements*' values)		✓		✓		✓			✓	
			Vendor-specific attributes	`VNFD.vdu.configurableProperties VNFD.configurableProperties VNFD.modifiableAttributes`	possibly (depends on the attribute's semantics)	possibly (depends on the attr. semantics)	Vendor of the MF and its underlying VNFD, or Operator	provided from a predefined MF-level *Slicing Entity Configuration Profile*, or from a dedicated service		?		?		?			?	
		`CreateVnf Request` [88]	`vnfInstanceName`	no	yes	Operator	generated based, for example, on the MF type, `ServiceProfile/SliceProfile`'s origin, etc.		✓		✓			✓	✓		
			`metadata`	possibly (depends on the attribute's semantics)	possibly (depends on the attr. semantics)	Vendor of the MF and its underlying VNFD, or Operator	provided from a predefined MF-level *Slicing Entity Configuration Profile*, or from a dedicated service		?		?		?			?	
		`Instantiate VnfRequest` [88]	`flavourId instantiationLevelId targetScaleLevelInfo`	yes, (but may not reflect the Slice/ Subnet design objectives)	yes	*Slicing Entity Configuration Profiles*, declaring achievable *SLS-satisfaction Capabilities*' values for Deployment Flavor and Instantiation Level	provided from a predefined MF-level *Slicing Entity Configuration Profile* by matching the declared *SLS-satisfaction Capabilities* with the SLS *Requirements*		✓		✓		✓			✓	
			`extVirtualLinks`	no	yes, if needed	NSD of the NS which includes this VNF	generated based on the topology in the abstract NST/NSST combined with the selected, mapped *Slicing Entities*		✓		✓			✓	✓		
	Application-level attributes	**`Function_`** [202] and the inheriting **`Managed Function`** [80]	`supportedPerfMetricGroups`	yes (but may not correspond to input SLS *Requirements*)	possibly (e.g. add custom KPIs)	SLA agreement in CSMF, SLS *Requirements* mapping to the corresponding KPIs	determined based on the input `ServiceProfile/ SliceProfile` SLS *Requirements* statically mapped to corresponding KPIs which monitor their fulfillment			✓	✓				✓		✓	
			`thresholdMonitor. thresholdInfoList ([..] thresholdValue, hysteresis)`	yes (but may not correspond to input SLS *Requirements*)	yes	input `ServiceProfile/ SliceProfile` *Requirements*, predefined hysteresis	input `ServiceProfile/SliceProfile` *Requirements* values statically mapped to the corresponding KPIs and measurements thresholds / hysteresis templates			✓	✓				✓		✓	
	Type-specific MF attributes [32]	**`GNBCUCP Function`** [32]	`plmnId`	possibly	yes, possibly	OSS/NSMF/NSSMF-level list of supported PLMNs	extracted from the input `ServiceProfile/SliceProfile` and confronted against the OSS/NSMF/NSSMF-level supported PLMNs			✓	✓				✓		✓	
			`configurable5QISetRef dynamic5QISetRef`	yes, (but may not reflect the Slice/Subnet design objectives)	yes, possibly	achievable 5QI values bound to each MF-level *Slicing Entity Configuration Profile*	supported 5QI values static mapping to *Slicing Entity Configuration Profile* selected by *Capabilities'* values matched to `ServiceProfile/SliceProfile` *Requirements*			✓		✓				✓	✓	
		`SMFFunction` [32]	`nRTAClist, taiList, taiRangeList`	no	yes	`ServiceProfile/SliceProfile` *Requirements* and the OSS/NSMF/NSSMF/NFMF repository of functions	input `ServiceProfile/SliceProfile`'s value (`coverageAreaTAList`) matched against the coverage of gNB-DUs to be supported by this SMF			✓		✓			✓		✓	
			`sNSSAIList`	no	yes	`ServiceProfile/SliceProfile` input *Requirements*	input `ServiceProfile/SliceProfile`'s values (combined SST/SD) confronted against the values supported in target environment (NSMF/NSSMF-level defined)			✓	✓				✓			✓
			`configurable5QISetRef dynamic5QISetRef`	yes, (but may not reflect Slice/Subnet design objectives)	yes, possibly	achievable 5QI values bound to each *Slicing Entity Configuration Profile*	supported 5QI values static mapping to *Slicing Entity Configuration Profile* selected by *Capabilities'* values matched to `ServiceProfile/SliceProfile` *Requirements*			✓		✓	✓			✓	✓	
Network Slice Subnet (Subnet)	**Network Service (NS)**-level attributes **`InstantiateNsRequest`** [89]	**`NSD`**[90] **`NsVirtual LinkDesc`**	`VirtualLinkDf.qos: (latency, packetDelayVariation, packetLossRatio)`	yes	possibly	Vendor of the Subnet and its underlying NSD or Slice/Subnet designer	provided from one of the predefined Subnet-level *Slicing Entity Configuration Profile* matched against the incoming, decomposed *Requirements*' values			✓		✓			✓		✓	
		`CreateNs Request` [89]	`nsName`	no	yes	Slice/Subnet designer	generated based, for example, on the Subnet type, `ServiceProfile/SliceProfile`'s origin, etc.			✓		✓			✓	✓		
			`nsflavourId nsInstantiationLevelId targetNsScaleLevelInfo`	yes, (may not reflect the Slice/ Subnet design objectives)	yes	*Slicing Entity Configuration Profiles*, declaring achievable *SLS-satisfaction Capabilities*' values for Deployment Flavor and Instantiation Level	provided from a predefined Subnet-level *Slicing Entity Configuration Profile* by matching the declared *SLS-satisfaction Capabilities'* values with the incoming SLS *Requirements*' values			✓		✓			✓		✓	
			`startTime`	yes, (now)	possibly	CSMF-level SLA agreement of the CSI	extracted from the CSI's SLA agreement in CSMF			✓		✓			✓		✓	
			`additionalParamsForNs`	possibly (depends on the attribute's semantics)	possibly (depends on the attribute's semantics)	Vendor of the Subnet and its underlying NSD, or Slice/Subnet designer	provided from a predefined Subnet-level *Slicing Entity Configuration Profile* or from a dedicated service			?		?			?		?	
		Existing VNF/NS instances to be **reused**	`vnfInstanceData (vnfInstanceId, vnfProfileId) nestedNsInstanceData (nestedNsInstanceId, nsProfileId)`	no	yes	info about VNF/NS instances underlying the *Runtime Slicing Entities* to reuse, NFVO API [89]	determined by *Requirements-Capabilities matching* and *selection* of the best fulfilling *Runtime Slicing Entities* (described in Subsections 4.3.2, 4.3.3 and 4.3.5)			✓		✓			✓		✓	
		Configuration for **new VNF/NS instances** to be spawned	`locationConstraints nestedNsLocationConstraints`	yes, (no constraints) (may not reflect the Slice/ Subnet design objectives)	possibly	coverage area requested in `ServiceProfile /SliceProfile`, registry of VIMs or PoPs in NFVO	determined by assigning the selected *Slicing Entities* to the closest available PoP according the requested coverage (`ServiceProfile`'s `coverageArea`, `SliceProfile`'s `coverageAreaTaList`)			✓		✓			✓		✓	
			`additionalParamsForVnf additionalParamsForNestedNs`	possibly (depends on the attribute's semantics)	possibly (depends on the attribute's semantics)	Vendor of the Subnet and its nested VNFs/NSes, or Slice/Subnet designer	provided from a predefined Subnet-level *Slicing Entity Configuration Profile* or from a dedicated service			?		?			?		?	
		Multi-Site Connectivity Service (MSCS) [83] for DCI to be **established** or **reused**	`wanConnectionData` – `nsVirtualLink` or `vnfVirtualLink`, – `protocolData` (`mscsInfo` – for existing MSCS instance to be reused, `mscsConfigData` – to create new)	yes, (no DCI needed)	yes, if DCI needed	predefined Subnet-level *Slicing Entity Configuration Profile*, WIM API [82] to access existing MSCSes	need for DCI if Managed Link binds *Slicing Entities* in different PoPs, characteristics of existing MSCSs matched to SLS *Requirements* against Managed Link, new MSCS provisioning data from Subnet-level *Slicing Entity Configuration Profile*			✓		✓	✓		✓		✓	
	Subnet-level attributes Standardized	**`Sub Network`** [80] (instance, which will contain this Subnet)	`supportedPerfMetricGroups`	yes (may not correspond to input SLS *Requirements*)	possibly (e.g. add KPIs for SLS *Requirements*)	SLS *Requirements* mapping to corresponding KPIs and performance metrics	determined based on the input `ServiceProfile/ SliceProfile` SLS *Requirements* mapped to corresponding KPIs which monitor their fulfillment			✓	✓				✓		✓	
			`thresholdMonitor. thresholdInfoList ([..] thresholdValue, hysteresis)`	yes (may not correspond to input SLS *Requirements*' values)	yes	`ServiceProfile/ SliceProfile` values, predefined thresholds or hysteresis templates	extracted from the input `ServiceProfile/SliceProfile` *Requirements* and mapped to the corresponding thresholds or hysteresis templates			✓	✓				✓		✓	
		`Network Slice Subnet` [32]	`managedFunctionRef networkSliceSubnetRef epTransportRef`	no	no	NSSMF's operational registry	no need (references should be established by the NSSMF orchestrator when the internal *Slicing Entities* are operational)			✓		✓			✓			✓
		Vendor-specific	(Vendor-specific)	possibly (depends on the attribute's semantics)	possibly (depends on the attribute's semantics)	Vendor of the Subnet and its nested VNFs/NSes, or Slice/Subnet designer	provided from a predefined Subnet-level *Slicing Entity Configuration Profile* or from a dedicated service			✓		?			?		?	
Managed Link (ML)																		

Finally, each attribute has been categorized according to the following classification criteria:

A. **management layer** to which it applies:
   - "platform" – pertains to the resources in the (virtualized) deployment platform and their organization. Deciding on the values for these attributes may determine the **footprint** of resources consumed by a particular *Slicing Entity* which, consequently, influences the consumption of resources by the designed NSI/NSSI perceived as a whole
   - "virtualized entity" - relates to the unit underlying the *Slicing Entity* in the NFV-MANO layer
   - "application" - relates to the specific role of a given *Slicing Entity* in the telco network and its configuration within the Slice(s)

B. **dependency** on the configuration of other *Slicing Entities*:
   - "independent" – value can be decided without analyzing configurations of other *Slicing Entities*
   - "dependent" - may indicate a more complicated way of determining the attribute's value, e.g. the need of waiting until the values for the dependent attributes are set

C. **impact** (even potential) of attribute's value on the *Slicing Entity's Capabilities* and their values:
   - "not impactful" – value of such attribute does not influence *Capabilities*
   - "impactful" – value of such attribute **influences *Capabilities* and their values**. Due to this fact, sensible, coherent and tested combinations of values for such attributes should be predefined in *Configuration Profiles*, carefully determined by the *Slicing Entity*'s Vendor or the Operator, in order to enable their easy and credible determination during the design process, based on the values of the offered *Capabilities*

D. **Slice-relevancy**
   - "Slice-agnostic" – value for such attribute is required for *Slicing Entity*'s **basic operation**, **even without any Slices assigned to it**.
   - "Slice-influenced" – determination of values for such attributes is also a prerequisite for *Slicing Entity*'s basic operation, but deciding on its concretized value might be **influenced by the SLS requirements posed against the designed NSI/NSSI**
   - "Slice-specific" - value for attribute is not required for basic operation (instance can be spawned without it), but it must be provided **to make the *Slicing Entity* serve a particular Slice**

   "Slice-agnostic" and "Slice-influenced" attributes are also referred to as **"bootstrap" attributes**, as they need to be supplied with concretized values to in order to make a given *Slicing Entity* operational.

An excerpt of the classification[108], for *Slicing Entities* representing MFs and Subnets, is presented in Table 5.

Analysis of the study results enables drawing several conclusions regarding the design process and its expected outcomes. Among others, the elaborated categorization allows inferring what information items' equivalence classes need to be contained in the Slice/Subnet *Instance Descriptor* to make the utilized internal *Slicing Entities fully concretized* depending on their current concretization level[109]. This is schematically presented in Figure 26 and commented in the following paragraph.

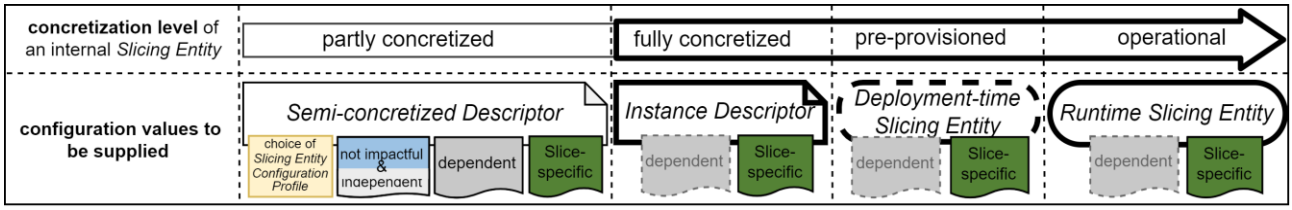


Figure 26. Groups of configurations values (labeled with categories as defined in Table 5) to be included in the finalized design outcomes to fully configure the internal *Slicing Entities* for instantiation and operation in the designed Slice/Subnet.

For internal *Slicing Entities* in the form of *Semi-concretized Descriptors*, the Slice/Subnet Instance Descriptor needs to include the choice of the initial *Configuration Profile* to be utilized at its deployment, values for the

[108] Full content of the elaborated study, including coverage for *Slicing Entities* representing MFs, Subnets and MLs, is provided as a supplementary material of [30] available under the following link: https://doi.org/10.1109/COMST.2023.3341249/mm1.

[109] *Abstract Templates* are not considered in Figure 26, as they require prior contextualization with not abstract *Slicing Entities*, which only then can be configured. Contextualization of *Abstract Templates* is described in Subsections 4.3.2-4.3.5.

"dependent" configuration attributes and for the rest of attributes that are not contained in *Configuration Profiles* – these are the attributes that are categorized as "not impactful" & "independent". In the case of *fully concretized Slicing Entities*, only a narrowed subset of configurations needs to be conveyed. This is due to the fact that one of their *Configuration Profile* is already selected and the configurations required for their basic operation are already specified[110]. Consequently, only values for the attributes classified as "Slice-specific" need to be conveyed and, potentially, for the configuration attributes classified as "dependent" (grayed-out in Figure 26). It is because while the values for "Slice-specific" attributes are needed to configure a *fully concretized Slicing Entity* to serve the designed NSI/NSSI (besides the ones already served by such a *Slicing Entity*), values for the "dependent" attributes shall be delivered if there is a need to supply information about a dependent *Slicing Entities* that will be newly spawned in the designed NSI/NSSI.

Furthermore, the study outcomes allow for informed reasoning about the placement of the configuration attributes in the proposed structure of the Slice and Subnet *Instance Descriptors* (presented in the following Subsections 3.3.3 and 3.3.5), about the need of determining values for these attributes and potential sources to be examined (columns A-C), as well as the methods of their automated determination and delivery in an automated design process (column D and Chapter 4). Due to their significance, the results of this study are frequently referred in the remaining content of this thesis.

3.3.3. Determination of information items pertaining to the designed NSI/NSSI as a whole

The previous Subsection has been concentrated on the determination of information items that pertain to the internal *Slicing Entities* of the designed NSI/NSSI, addressing aspect 1 of the Slice/Subnet Instance Descriptor's role. However, in order to cover aspect 2, it also needs to contain a number of information items that concern the **group** of internal *Slicing Entities* **perceived as a single, encapsulated unit**. Essentially, to address aspect 2.1, a Slice/Subnet *Instance Descriptor* needs to be a fully-fledged *Slicing Entity* and, as such, it must be possible to clearly assess what level of SLS satisfaction may be achieved if an NSI/NSSI is created out of it, as well as what its offered functionalities and exposed endpoints are. Thus, the *Instance Descriptor* must declare *Capabilities* of the designed Slice/Subnet as well as its *Configuration Profiles*, which configure it to expose particular values of the offered *Capabilities*.

Due to the fact that a Slice/Subnet is essentially a composite of its internal *Slicing Entities*, the *Capabilities* of the Slice/Subnet *Instance Descriptor* actually constitute ***aggregated Capabilities*** [111] offered by its encapsulated *Slicing Entities*. In this context it shall be noted that the aggregated *Capabilities*' values are possible to be determined only if all the internal *Slicing Entities* expose their concretized *Capabilities*' values - this, in turn, is only achievable when their "impactful" configuration attributes have decided values, which is assured by the choice of their *Configuration Profiles*[112]. As such, a gathering of *Configuration Profiles* chosen for all internal *Slicing Entities* (potentially supplemented with values for some of the Slice/Subnet-level configuration attributes) effectively represents a particular ***Configuration Profile*** **of the Slice/Subnet** ***Instance Descriptor***. At least one **"primary"** *Configuration Profile* that configures internals to fulfill the SLS *Requirement*s which triggered the design process should be available. Additionally, **alternative** Slice/Subnet-level *Configuration Profiles* that point to different *Configuration Profiles* of the internal *Slicing Entities* and contain different settings for the Slice/Subnet-level attributes (discussed in the following paragraph) may be defined in the Slice/Subnet Instance Descriptor as well. Obviously, every specified *Configuration Profile* must have a calculated set of corresponding *Capabilities*' **values** that need to be included in the Slice/Subnet *Instance Descriptor* as well.

On top of the already mentioned information items, the created *Instance Descriptors* should also cover aspect 2.2, by containing attributes that pertain to the performance supervision of the designed NSI/NSSI, which are referred to as **Slice/Subnet-level SLS-assurance configurations**. Such attributes include the definitions of the

[110] In the study, it is assumed that the *fully concretized Slicing Entities* are utilized in the designed NSI/NSSI "as is". However, their reconfiguration is possible. Then, values for the changed attributes should be included in the Slice/Subnet *Instance Descriptor*, e.g. choice of another *Configuration Profile*.

[111] The methods of performing aggregation depend on the data type and semantics of *Capabilities* being aggregated – see upcoming Subsection 3.5.

[112] This is determined in the *selection* activity discussed in the upcoming Subsections 4.3.3 and 4.3.5.

Slice/Subnet-level KPIs, their thresholds, hysteresis of alarms generation, as well as specification of policies governing the SLS-assurance mechanisms in the utilized Orchestator. Based on such inputs, SLS-assurance policy *Configuration Profiles* transition rules can be also defined in the Slice/Subnet *Instance Descriptor*, which will enable automated adaptability of the designed NSI/NSSI to the conditions experienced during runtime.

3.3.4. Summary of information items to be contained in a Slice/Subnet Instance Descriptor

In summary, the identified equivalence classes of information items that shall be contained in a Slice/Subnet *Instance Descriptor* are presented in Figure 27 and enumerated below:

a. *Aggregated Capabilities' attributes*
b. Values of *aggregated Capabilities' attributes*, for each *Configuration Profile*
c. Internal *Slicing Entities*' identification, topology and geolocation
d. Slice/Subnet-level *Configuration Profiles*, including:
   e. Choice of *Configuration Profiles* for the internal *Slicing Entities*, which specify concretized values for their "impactful" attributes
   f. Values for "not impactful" attributes that are related to a given *Configuration Profile*
g. Slice/Subnet-level SLS-assurance configurations
h. Values for attributes that are not impacted by changing *Configuration Profiles*

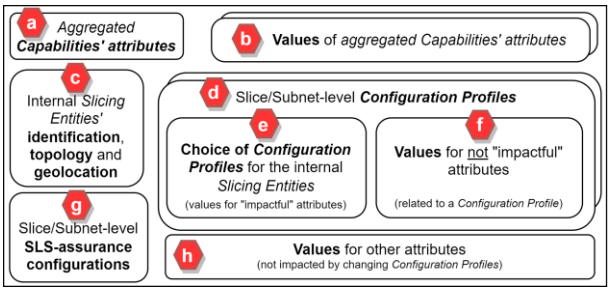


Figure 27. Groups of information items to be conveyed by a standards-aligned Slice/Subnet *Instance Descriptor*.

3.3.5. Layered distribution of information items among 3GPP and NFV-MANO Descriptors

As discussed in Subsection 2.3.5, the currently standardized Network Slicing orchestration stack is composed of several blocks devoted to the management of entities comprising an NSI/NSSI at different layers. Each of such systems requires dedicated input for its operation. Thus, not only Slice/Subnet *Instance Descriptors* for the 3GPP NSMF/NSSMF blocks need to be produced as the outcome of the NSI/NSSI design process, but also **fully concretized design-time descriptions of the NFV-MANO entities** underlying the Slicing objects.

In this context, it shall be noticed that the assignment and organization of the compute, memory, storage and networking resources of the internal *Slicing Entities*, but also their geolocation and the topology of L2/L3-layer links, are aspects that are already covered by the information items contained in the models of the ETSI NFV-MANO Descriptors. Consequently, these information items do not need to be duplicated in the Subnet *Instance Descriptor* - it is sufficient that it **references an NSD** which, together with the internally referenced VNFDs, VLDs and nested NSDs, covers all the "platform-layer" and "virtualized entity-layer" aspects (see column E, Table 5) of the designed Subnet, its internal MFs, MLs and any nested Subnets, respectively. Taking these facts into account, a conclusion is drawn that Slice/Subnet *Instance Descriptors* shall cover the rest of the identified information items that are not covered by the underlying NFV-MANO Descriptors. These aspects include the "**application-layer**" information items (see column E, Table 5) for internal *Slicing Entities* pertaining to the MFs (deployed inside of VNFs) and their configurations, the topology of application-layer MLs between the internals, specification of the Slice/Subnet-level KPIs and SLS-assurance policies as well as the *aggregated Capabilities* and their values.

It shall be also noted that in order to assure flexible composition of the Slice/Subnet structure during the design process, the **NSDs** corresponding to the decided Subnets' structure typically also **need to be newly generated** as a design outcome, along the Subnet *Instance Descriptor*. Such a conclusion is drawn because utilization of any predefined NSD does not allow for dynamic specification of the identity and topology of the internal *Slicing Entities* - NSD model requires explicit references to particular internals, making it static. The

topology and identity of *Slicing Entities* to be utilized in the design NSI/NSSI, however, are not known a priori, but are dynamically decided during the design process. As such, generation of NSDs reflecting the design decisions in the NFV-MANO layer is needed. Moreover, to make the generated NSD *fully concretized*, the `CreateNsRequest` and `InstantiateNsRequest` data structures filled with instance-specific configuration for the underlying NFV-MANO entities need to be produced as a design outcome as well.

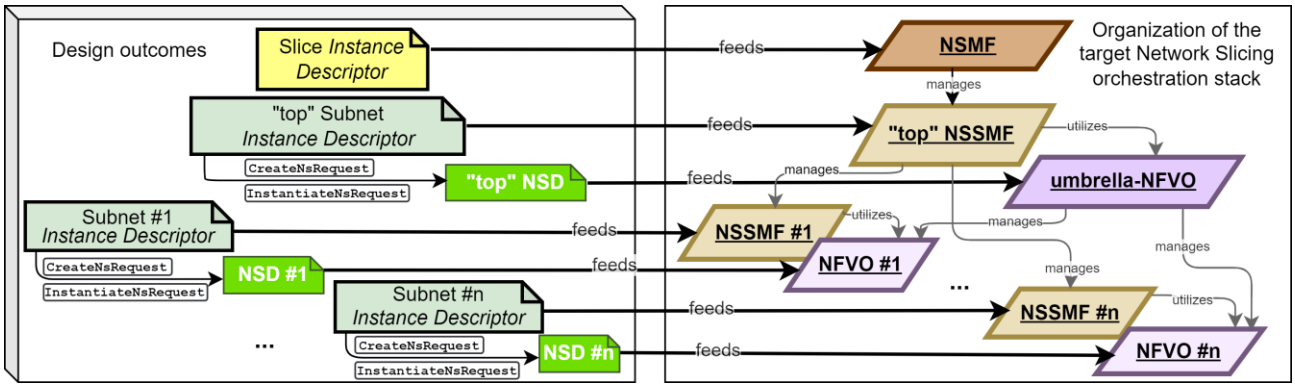


Figure 28. Example of the Network Slice design outcomes produced as multiple *Instance Descriptors* that feed the orchestration blocks in a particular organization of the 3GPP-ETSI compliant Network Slicing orchestration stack. For clarity reasons, design-time descriptions of the *lowest-level Slicing Entities* (MFs, MLs) and their underlying ETSI NFV-MANO counterparts (VNFDs, VLDs), are not depicted.

Figure 28 illustrates the discussed division, in which the outcome of the Slice design process[113] is composed of a set of *Instance Descriptors*, NSDs and their concretizing data structures, as shown on the left side. The created Slice *Instance Descriptor* feeds the NSMF, the "top" Subnet *Instance Descriptor*, the "top" NSD and its concretizing data structures feed the "top" NSSMF and "umbrella-NFVO"[114], respectively. Analogically, *Instance Descriptors* of the internal Subnets, their corresponding NSDs and concretizing data structures are used by the NSSMFs and NFVOs in the particular domains (NSSMF#1..n, NFVO#1..n).

3.3.6. Proposed structure of 3GPP-aligned Slice/Subnet *Instance Descriptors*

Based on the inference provided in the previous Subsections, a **generic, high-level structure** of a Slice and Subnet *Instance Descriptors* is proposed, catered to be compliant with the 3GPP runtime management models defined in the 5G NRM and the ETSI NFV-MANO Descriptors. By following the encapsulation principle proposed as a part of the *Slicing Entities* taxonomy, the information items contained in the Slice and Subnet *Instance Descriptors* are proposed to be organized into hidden internals and exposed *Capabilities*, as schematically depicted in Figure 29 – this Figure utilizes the labels of information items equivalence classes as assigned in Figure 26 and Figure 27.

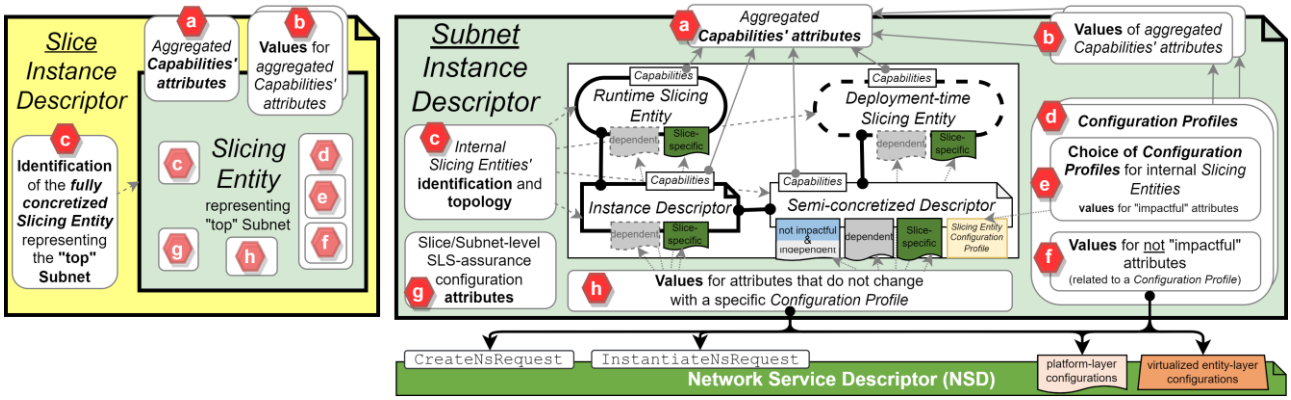


Figure 29. High-level structure of 3GPP-compliant Network Slice/Subnet *Instance Descriptors*, with groups of information items labeled as in Figure 26 and Figure 27.

In the case of both Slice and Subnet *Instance Descriptors*, the exposed information comprises the *aggregated Capabilities attributes* and their declared values (equivalence classes "a" and "b") per each of the offered

[113] In the example in Figure 28, it is assumed that the design process for the entire NSI, including all its NSSIs, is performed centrally inside of NSMF. However, the particular NSSIs may be designed independently inside of the particular domains.

[114] A higher-level, cross-domain NFVO is frequently referred to as an "umbrella-NFVO" in the ETSI documents, e.g. [203].

Configuration Profile. The internals of the proposed Slice and Subnet *Instance Descriptor* contain the identification of the internal *Slicing Entities* and the application-layer topology between them (equivalence class "c"). As depicted in Figure 29, Subnet *Instance Descriptor* may reference multiple *Slicing Entities* that represent the structure of the designed Subnet: its nested Subnets, MFs and topology of MLs. On the other hand, a Slice *Instance Descriptor* only points to a single *fully concretized Slicing Entity* representing the "top" Subnet. It is also worth noting that in a Slice *Instance Descriptor*, the *Capabilities attributes* and their corresponding values (equivalence classes "a" and "b") are the ones offered by the internal *Slicing Entity* that represents the "top" Subnet. Such construction of the Slice *Instance Descriptor* allows for full alignment of the proposed *Instance Descriptors*' models to the *1-to-1* relationship between the standardized `NetworkSlice` and `NetworkSliceSubnet` runtime management models defined in 5G NRM.

Besides the aforementioned information items, a Subnet *Instance Descriptor* contains the Slice/Subnet-level SLS-assurance attributes (equivalence class "g") and configuration values for all the internal *Slicing Entities* (equivalence classes "e" and "f"). It also includes any remaining Slice/Subnet-level configuration attributes (equivalence class "h") required to make the design outcomes *fully concretized*. The coherent sets of values for the "impactful" configuration attributes are gathered into Slice/Subnet-level *Configuration Profiles* (equivalence class "e"). By applying a particular *Configuration Profile*, the Slice/Subnet *Instance Descriptor*, just like any other *Slicing Entity*, exhibits particular values of the *aggregated Capabilities* (equivalence classes "a" and "b").

As depicted in Figure 29, based on the selection of internal *Slicing Entities*, their topology and configurations, an NSD accompanying the Subnet *Instance Descriptor* shall be generated to reflect these design decisions at the level of the NFV-MANO entities. Based on the values for the "platform-layer" and "virtualized entity-layer" configuration attributes contained in the Subnet *Instance Descriptor*, the data structures contextualizing the generated NSD (`CreateNsRequest`, `InstantiateNsRequest`) can be filled with values. Such structuralization enables fully automated, unattended spawning of the underlying NS instance, which, in turn, spawns all the instances of VNFs, nested NSs and VLs underlying the internal *Slicing Entities*.

As a final remark it shall be noted that the proposed structure of Slice and Subnet Descriptors has been determined in a generic manner, to be fully aligned with the standardized specifications in force. Obviously, the ultimate models of such Descriptors in a given Orchestrator will depend on the level of its standards-compliancy and specific architectural decisions.

3.4. Modeling of individual SLS attributes and their handling

In the discussions until now, the SLS *Requirements* contained in the `ServiceProfile`/`SliceProfile` data structures have been mainly mentioned collectively, as sets of "some" attributes expressing the Customer's expectations against the NSI/NSSI to be allocated. The *Slicing Entity Capabilities* have been presented in a similar way – as sets of "some" attributes characterizing a given *Slicing Entity*. However, the definition of the design process provided in Subsection 3.2.1 indicates that both *Requirements* and *Capabilities* are subjects of processing in various design activities during its course. To enable unattended execution of these activities, **detailed knowledge about the semantics and methods of handling each individual attribute** contained in the processed sets of *Requirements*/*Capabilities* needs to be captured in appropriately structured models that can be effectively exploited in an automated process.

In Subsection 3.1.2 it has been argued that the *attributes* exposed as *Capabilities* of a given *Slicing Entity* should semantically correspond to the *Requirements attributes* that may be posed against it in the course of the design process. Such assumption leads to the proposition of a modeling approach in which the *attributes* representing *Requirements* and *Capabilities* are covered with a single, generic model. However, although a given *Requirement*/*Capability attribute* may have the same name, data type and semantics, the *attribute* model needs to take into account that the **methods of its processing in various design activities** differs, depending on the represented perspective.

More specifically, the model of an *attribute* needs to specify the method of **comparing** its value as a *Requirement* with the value of a *Capability*. It also must be known what level of *Requirement* fulfillment is represented by the particular comparison results and which of them can be considered as a **match**.

Furthermore, to enable differentiating comparison results to the ones representing better or worse match, a **scoring** method shall be specified as well. Next, the attribute's model needs to specify how its value as a *Requirement* **accumulates**, allowing computation of the total value of a given *Requirement* posed against a single *Slicing Entity* that handles multiple sets of *Requirements* assigned to it. Furthermore, the attribute's model must also specify how its value as a *Capability* is "**consumed**" by the accumulated *Requirements*. Obviously, specification of the mentioned methods for a given *attribute* depends on its data type and operational meaning. Typically, the methods of handling a given *Capability*/*Requirement attribute* constitute simple arithmetical operations. However, support for more advanced transformations might be needed in the case of *attributes* with complex data types and advanced operational semantics.

As an example, specification of methods for automated processing of the `dLThptPerSlice` ***attribute*** is as follows. Since the data type of this attribute is a real number, the comparison method of its values is straightforward - the *Capability* value must be higher or equal than the value of the posed *Requirement* in order to be evaluated as a match, and the higher the *Capability* value is, the higher matching score is computed. If multiple sets of *Requirements* are assigned to a single *Slicing Entity,* then the accumulated value of the `dLThptPerSlice` *Requirement* assigned to it constitutes a sum of the posed ones. The value of the `dLThptPerSlice` *Capability* is deducted by the accumulated *Requirements* value.

Once the knowledge about the enumerated processing methods is defined for all *Requirement*/*Capability attributes* supported in the Orchestrator[115], their automated processing in a number of design activities becomes possible, as described in the upcoming Chapter 4. There are, however, two other transformations of the SLS attributes that, due to their crucial role in the design process and its automation, deserve dedicated coverage. In the following Section, modeling of ***Capabilities aggregation*** and ***Requirements decomposition*** within the internal structure of the designed NSI/NSSI is elaborated in detail.

3.5. Modeling of SLS handling in the Slice/Subnet internal structure

Another piece of knowledge that needs to be captured and modeled in order to be leveraged in an automated design process relates to the **internal structure** of Slice/Subnet that shall be designed to satisfy the particular *Requirements* posed by the Customer. Typically, the not fully concretized design-time descriptions utilized in the currently prevailing imperative Orchestrators (e.g. customized kinds of Descriptors or Templates) already allow for conveying the knowledge about the topology of internal building blocks comprising a Slice/Subnet, their required types and cardinalities, e.g. by adopting a similar modeling approach as utilized in the constructs of the ETSI NSD model, which have been already elaborated in Subsection 2.2.3. For this reason, detailed elaboration of these aspects is skipped. However, in order to serve in an automated design process, such design-time descriptions need to become fully-fledged *Design-time Slicing Entities* and, as such, typically they need to be **augmented with additional knowledge that will support design automation**. This information relates to the aspects not covered by the NFV-MANO models, namely a comprehensive handling of SLS, both from the perspective of *Capabilities* and *Requirements*, as described in the following Subsections.

3.5.1. Modeling of *Requirements* decomposition

To support automated design process, the *not fully concretized Slicing Entities* representing a Slice/Subnet (i.e. *Abstract Templates* and *Semi-concretized Descriptors*) need to convey the information on how each of the high-level *Requirements* present in the input `ServiceProfile`/`SliceProfile`, which pertain to the Slice/Subnet as a whole, is distributed among the Slice/Subnet internals. More specifically, it needs to be defined how to **decompose** the high-level SLS *Requirements* into lower-level *Requirements*, which with their abstraction level and semantics will be possible to be matched to the *Capabilities* of the *lower-level Slicing Entities* considered for utilization in the designed NSI/NSSI.

As an example, if a given NSST shall be considered for automated design of a Subnet satisfying a particular value of the `dLLatency` *Requirement* (e.g. 20ms), then it needs to expose the `dLLatency` *Capability* with a value in a matching range (e.g. 10ms < v <= 25ms). Furthermore, such NSST must specify how this *Requirement* and

[115] As justified in Subsection 2.3.2, a given Orchestrator should clearly indicate which SLS attributes are supported, as the set of standardized attributes is broad and composed mainly of optional ones. Additionally, there might be vendor-specific attributes defined as well.

its value shall be decomposed into lower-level *Requirements* to be posed against internal *Slicing Entities* that handle the user-plane traffic, namely the gNB-DU and gNB-CU-UP MFs as well as the F1-U and F2-U MLs connecting them (see Figure 6 in Subsection 2.1.3). Depending on the features of the *lower-level Slicing Entities* supported by the Orchestrator, the transformed *Requirements* may pertain to their low-level settings that need to be enforced in order to achieve the demanded latency value, e.g. allocation of PRBs fraction[8], scheduling traffic with a particular priority, configuration of protocol's frame size, etc. If the `dLLatency` *Requirement* value is stringent (e.g. 5ms) then another NSST that assumes the presence of UPF MF in its topology may be considered, providing it exposes a matching *Capability* value (e.g. 5ms <= v <= 10ms) and defines transformation of such *Requirement* into low-level configurations of internal *Slicing Entities* that will be capable of satisfying it.

Determining what is the correspondence between the high-level *Requirements* and the lower-level ones is a challenging task, **requiring expert domain knowledge** to appropriately understand and interpret the higher and lower-level *Requirements*, as well as deep technical knowledge of how the particular *SLS-satisfaction Requirements* can be satisfied by the *Capabilities* offered by the given *Slicing Entities* in their *Configuration Profiles*. All this knowledge depends highly on the features of the utilized *Slicing Entities* and, as such, needs to be specified either by the *Slicing Entities*' Vendors or the Operator.

To provide appropriate means for expressing such knowledge, the author proposes a generic modeling construct of ***Requirement Decomposition Rules*** **(RDR)** that enable performing automated *Requirements* decomposition by leveraging a series of **generic transformations**. More specifically, the decomposition of a particular, single *Requirement attribute* consists of a series of transformations that involve its gradual ***translation***, ***assignment*** and ***distribution of its value***, as schematically depicted in Figure 30 below. This is valid in a generic case, despite the large diversity of *Requirements attributes* and their highly domain-specific particularities (i.e. a growing range of *SLS-satisfaction Requirements attributes* specified by 3GPP in the `ServiceProfile`/`SliceProfile` data structures, but also *functional* and *composability Requirements*), and therefore justifies proposition of a generic model. Obviously, each *Requirement* has its domain-specific decomposition behavior that determines the methods of its translation (e.g. into a single or multiple lower-level *Requirements*), assignment (e.g. to a single, lower-level *Slicing Entity*, or a group of them) and values distribution (e.g. proportional or sequential division of value, exact copy, etc.). Hence, in the following paragraphs, the mentioned transformations are elaborated in detail, providing several representative examples.

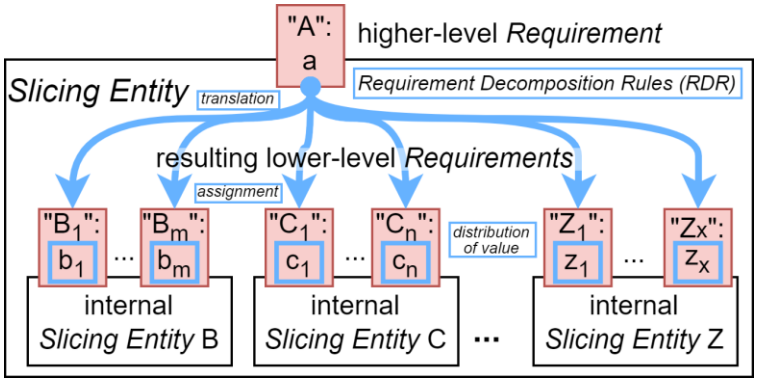


Figure 30. Graphical presentation of *Requirement Decomposition Rules* (RDR) transformations: translation, assignment and distribution of value.

The **translation** transformation aims at determining representation of a higher-level *Requirement attribute* by one or more lower-level *Requirement attributes* which reflect the semantics of the original attribute, but are expressed in terms of the lower-level characteristics. For example, the `coverageArea` *attribute* in the `ServiceProfile`, which according to the 5G NRM can be expressed by means of high-level geographical area units (e.g. name of a stadium, location of a certain enterprise site or by a set of geodetic shapes [158]), needs to be translated into the list of TAI codes[61] for the Core and RAN Subnets, expressed in the `coverageAreaTAList` attribute of `SliceProfile`. Then, the TAI list should be further translated into the lowest-level *Requirements* against the *Slicing Entities* representing gNB and other relevant MFs, which will be translatable to their configuration attributes. It is worth mentioning that in multiple cases, the translated lower-level *Requirement attribute* has the same name and semantics as the higher-level *Requirement attribute* but it pertains to the *lower-level Slicing Entity*[116].

[116] e.g. `dLLatency` is present both in `ServiceProfile` as well as in all variants of the lower-level `SliceProfile` specified in the 5G NRM: `TopSliceSubnetProfile`, `RANSliceSubnetProfile` and `CNSliceSubnetProfile`.

After translation, the derived attribute(s) representing the lower-level *Requirement(s)* need to be appropriately **assigned** to the relevant *Slicing Entity(ies)* which, by their offered *Capabilities*, contribute to the fulfillment of a given *Requirement*. For example, after translating the `dLLatency` ***SLS-satisfaction Requirement*** contained in the `RanSliceSubnetProfile`, the lower-level `dLLatency` ***Requirement*** shall be assigned to the gNB-DU and gNB-CU-UP MFs and the F1-U, F2-U MLs between them, because these are the *Slicing Entities* which participate in the user-plane traffic exchange. Conversely, the `dLLatency` ***Requirement*** should not be assigned to the gNB-CU-CP as it does not influence the user traffic latency.

Finally, the **distribution** of a higher-level *Requirement* **value** into the values of its lower-level counterparts needs to be performed. Taking the example from the previous paragraphs, the 30ms value of the `dLLatency` ***SLS-satisfaction Requirement*** in the `SliceProfile` needs to be split among the assigned *Slicing Entities* inside of the RAN Subnet. The strategy may be, for example, to distribute this value proportionally to the lower-level `dLLatency` ***Requirement*** posed against the *Slicing Entities* representing the gNB-DU, gNB-CU-UP MFs and the F1-U, F2-U MLs. It shall be noted, though, that the fulfillment of a higher-level SLS value might be realized in multiple variants of the lower-level SLS values. As an example, the end-to-end Slice-level `dLLatency` budget demanded in the input `ServiceProfile` may be statically split among RAN, Transport and Core domains or, in the case of RAN underperformance, the overall latency budget may be satisfied by posing more stringent values on the Transport or Core domains[117]. As such, the decision on the distribution of a particular *Requirement*'s value may be either defined statically (e.g. split proportions) or determined dynamically, steered by configured policies evaluated at the moment of performing *Requirements decomposition*. Such policies may take into account a variety of aspects, e.g. business agreements with the Subnet providers in the particular domains, or the current, historical and predicted load of the deployment environments and the considered *Runtime Slicing Entities*, etc.

While the examples provided above were concentrated on *SLS-satisfaction requirements*, the discussed transformations are equally applicable to decompose the *functional* or *composability Requirement*s. For example, the composability *Requirement* against a particular RAN Subnet may be to expose an `EP_Transport` endpoint for the N2 reference point. In order to meet this *Requirement* on the level of Subnet internals, such *Requirement* gets decomposed by translating it into the lower-level *Requirement* to expose an `EP_RP` endpoint for the N2 reference point (`EP_N2`), which gets assigned to the *Slicing Entity* representing gNB-CU-CP MF. Similarly, a high-level functional *Requirement* against a Core Subnet to deliver "UE location exposure" functionality may be translated to two functional *Requirement*s to expose the `Namf_Location` and `Nnef_Location` Network Function Services [68], assigned to the *Slicing Entities* representing the AMF and NEF MFs respectively.

3.5.2. Modeling of *Capabilities* aggregation

As already broadly elaborated in Subsection 3.1.2, each *Design-time Slicing Entity* prepared for automated processing needs to be equipped with a set of credibly defined *Capabilities*. But, as argued in Subsection 3.3.1, a Slice/Subnet is a virtual composite management entity that does not deliver any *Capabilities* by itself, as all of them are delivered by the internally contained, *lower-level Slicing Entities*. Hence, in order to know a particular set of *Capabilities* exposed by a *Slicing Entity* representing Slice/Subnet as a whole (both the *Design-time* and *Deployment-time*, as well as the *Runtime* ones), the knowledge about how the lower-level *Capabilities* are reflected in the higher-level ones needs to be captured in appropriate model that would allow its effective, automated processing.

In order to convey such knowledge in a generic case, a conceptual model of ***Capability Aggregation Rules*** **(CAR)** is proposed by the author. A CAR for a particular *higher-level Capability* comprises two (instead of three, like in the case of RDR) information items: ***source Capabilities identification*** and specification of the ***aggregation method***. Explicit *translation* of the *Capability* attribute like in the case of RDR, does not need to be specified, as it is implicitly defined by the identified source *Capabilities* and the target *Capability* for which the CAR is specified. The transformations modeled by CAR are graphically depicted in Figure 31. Depending on the semantics of a given *Capability*, there may be various *aggregation methods*, e.g. min, max, average,

[117] Such "SLS budget reshuffling" might also happen during NSI/NSSI operation, as a part of their SLS-assurance.

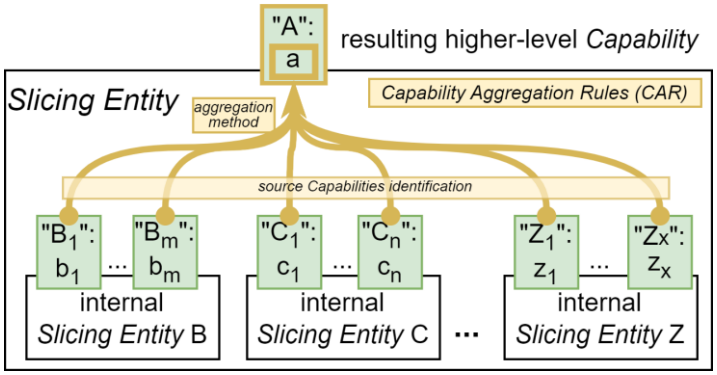


Figure 31. Graphical presentation of *Capability Aggregation Rules* (CAR) elements: source *Capabilities* identification and aggregation method.

product, etc. However, in the case of *Capabilities* with complex data types or when merging several *Capabilities* of different types into a single one, the aggregation method may be defined by a more complex, specific processing logic.

As an example, the value of the `dLLatency` *SLS-satisfaction Capability* to be advertised by a *Configuration Profile* of the designed RAN Subnet shall be aggregated by summing up the lower-level `dLLatency` *Capabilities*' values exposed by the *Configuration Profiles* of internal *Slicing Entities*, which participate in the user-plane traffic exchange (namely the gNB-DU and gNB-CU-UP MFs and the F1-U, F2-U MLs between them). In turn, the `dLLatency` to be advertised by the *Configuration Profile* of the end-to-end Slice ("top" Subnet) is aggregated by summing up the `dLLatency` values exposed by the *Configuration Profiles* of its contained Subnets. Or, when aggregating the `coverageArea` *Capability* on the level of the end-to-end Slice[118], the values of the `coverageAreaTaList` *Capability* offered by the internal *Slicing Entities* representing Subnets need to be transformed by a dedicated logic capable of determining the high-level geographical objects (e.g. geodetic 2D or 3D shapes standardized in [158], addresses, district names, etc.) based on the TAI codes[61] utilized by a particular Provider.

3.6. Modeling of operational and business objectives

The definition of the design process provided in Subsection 3.2.1 emphasized the need of aligning the design decisions to the objectives of the Provider that offers NSI/NSSI allocation to its Customers. Thus, as a final piece of knowledge to be modeled, the Operator's and Provider's **operational and business objectives** that may influence the decisions in the particular design activities need to be captured. Such objectives may pertain to optimization of a given operational goal, e.g. maximization of SLS fulfillment, minimization of resources utilization, etc. Optimization of some business objectives may be also considered, e.g. reduced cost, preferential treatment of premium customers, etc. On the technical level, such preferences may influence, for example, the selection of *Slicing Entity Configuration Profiles* which offer the best tradeoff between the level of SLS fulfillment and the actual (in the case of considered *Deployment-time* and *Runtime Slicing Entities*) or declared (in the case of *Design-time Slicing Entities*) footprint. It may also be promoted to reuse the already existing, sharable *Runtime Slicing Entities* in order to optimize the utilization of resources, providing they have enough capacity to accommodate the input *Requirements* along the other sets of *Requirements* already assigned to them.

While in a manually performed *imperative-style* design process it is the responsibility of the human Operator to make design decisions aligned with the Providers objectives, for the sake of organizing an automated design process this knowledge needs to be expressed with models that can be utilized by automated logic. Typically, simple configuration files containing sets of settings (potentially hierarchized) allow for successful modeling of such objectives in the majority of use cases. However, if more advanced reasoning logic is needed, then **policies** evaluated by a policy engine deliver powerful tooling allowing the expression of Operator's and Provider's objectives in a flexible way. This is achieved by defining a set of reasoning **rules** that dynamically determine the design decision in the particular circumstances. More about policies influencing the design process decisions is discussed in the upcoming Subsections 4.3.1 and 4.3.3.

3.7. Summary

In this Chapter, a range of original design-time concepts, definitions and constructs that enable modeling of the essential inputs and outputs of an automated design process have been introduced and explained. Importantly, all the proposed modeling concepts provide the added value of **representing the necessary**

[118] According to 5G NRM, the `coverageArea` in `ServiceProfile` can be expressed by formats assumed in the GSMA GST specification [35], where the "Area of service" can be expressed by means of geodetic shapes, civic addresses or identification of known locations (e.g. name of stadium, etc.).

pieces of expert operational and domain knowledge, laying the foundations for its exploitation in an automated design process that meets the requirements stated in Section 2.4.

Among others, it has been shown how the proposed concept of a *Slicing Entity* allowed for a **systematic classification** of the building blocks out of which an NSI/NSSI may be designed, which has not been elaborated in such an extent in the state of the art yet. In particular, the notion of a *Slicing Entity* encompasses a large variety of such building blocks by covering them all by a **single, uniform modeling abstraction**. Furthermore, the constructs of ***Capabilities*** and ***Configuration Profiles*** not only allow the Vendors and/or the Operator to deposit their knowledge about the essential characteristics of the utilized *Slicing Entities*, but it also enables perceiving them as logically **encapsulated** individuals, **abstracting away** their **internal complexity**. This virtue is of significant value for the organization of an automated Network Slice/Subnet design process, as it allows considering a potentially large set of heterogenous *Slicing Entities* based on the characteristics essential in the design process only, without the need of knowing their highly specific details.

Introduction of the mentioned concepts also facilitated formulation of precise definitions of the design process and its expected outcomes. It also allowed establishing the structure of Slice and Subnet *Instance Descriptors* that have not been defined in the standards until now. Finally, the other pieces of domain and operational knowledge necessary for conducting an unattended design process have been identified and appropriate modeling constructs enabling its representation have been proposed, including the conceptual models of *Requirement*/*Capability* attribute, *Requirement Decomposition Rules* (RDR), *Capability Aggregation Rules* (CAR), as well as generic models allowing expression of Operator's operational and Provider's business objectives.

Chapter 4

Conceptual framework of a Network Slice/Subnet design process and its automation

In this Chapter, the study of the Network Slice/Subnet design process is continued, with the focus on analyzing its course within an end-to-end allocation process. The analysis is performed by decomposing the definition of the design process provided in Subsection 3.2.1 into fine-grained, individual design activities, and by discovering the flows of control between them in a generic case. Based on the gained understanding, deep study of the processing logic in each of the identified design activities is conducted, with the aim of devising methods of their automation, and the ways of steering the design course in the particular operational circumstances. The conducted studies result in the proposition of a holistic, coherent and generic conceptual framework that puts foundations for the organization of a fully standards-aligned Network Slice/Subnet design process and its automation, serving as a reference for constructing Orchestrators supporting *declarative style* of NSI/NSSI allocation. Elaboration of the presented framework is among the key author's contributions, as currently, the course of the design process and methods of its automation are not specified by any standards and are not comprehensively covered in the state of the art (see Section 2.5).

The content of this Chapter is organized as follows. First, Section 4.1 discusses a number of introductory remarks regarding the assumptions and the presentation scope of the elaborated framework. Then, in Section 4.2, a high-level walkthrough of the design process organization is presented, introducing its inputs, typical as well as optional activities and flows. Next, in Sections 4.3-4.6, the logic and automation methods of the particular design activities are discussed one by one, starting with the inherent activities that define the internal structure of the designed NSI/NSSI and its high-level characteristics, up to the ones that finalize the design outcomes and execute post-design procedures related to their operational utilization by the Orchestrator. Finally, in Section 4.7, several aspects of end-to-end design process automation are discussed, including gradual rollout of automation mechanisms, desired level of end-to-end automation and technical aspects related to practical implementation of the presented framework.

4.1. Preliminary remarks

Before diving into the details of the elaborated framework, several preliminary remarks regarding the **scope** and **generality** of the content provided in this Section need to be highlighted. In its original version, the presented conceptual framework has been described in the author's major publication [30]. For this reason, while in Subsection 4.2 an overview of the framework's full extent is outlined, detailed study of design activities in this dissertation is provided for their selected subset that is essential for thesis statements verification - for an extensive coverage of aspects deliberately omitted in this work, the reader is kindly referred to [30].

The presented conceptual framework is generic and agnostic from the perspective of the provisioning style to be achieved through its implementation. As such, although in this thesis it is presented mainly in the context of achieving fully-automated, *declarative style* NSI/NSSI allocation, the provided descriptions can be leveraged for the organization of a semi-automated (assisted) or manual design process that can be leveraged in the *imperative style* solutions. Moreover, due to the *1-to-1* relationship between Slice and Subnet models explained in Subsection 2.3.3 and the recursive nature of a Subnet, the described framework is equally applicable for designing end-to-end Slices ("top" Subnets), high-level Subnets as well as their lower-

level, nested Subnets, regardless of the nesting level. Indeed, in the case of both Slice and Subnet, the nature of inputs to the design process is the same, the design activities are the same and the produced outcomes have a similar structure, as discussed in Subsection 3.3.5.

In a particular realization, implementation of a design process according to the presented framework will usually constitute an orderly workflow of interweaved or iterative combination of the described activities, their subset or a superset including vendor-specific extensions, not necessarily executed in the presented order or extent. It shall also be noted that over the execution of the design activities, multiple exceptional conditions may occur, impeding or totally blocking the execution of the design process. There may be various causes for such circumstances, including technical reasons (e.g. problems with connectivity to the utilized systems, etc.) or operational ones (e.g. lack of suitable *Slicing Entities* fulfilling the SLS *Requirements*, insufficient capacity of the target deployment environments, etc.). Successful execution of the entire design process requires their identification and appropriate handling at each of the described activities. Discussing all the possible failures and the ways of their handling is too broad to be conveyed comprehensively and is not in the scope of this thesis. Thus, in the provided descriptions, favorable environmental conditions and successful execution of the enumerated activities are assumed.

4.2. High-level overview

In this Section, a high-level overview of the conceptual framework of a Network Slice/Subnet design process is provided, by briefly describing its structure depicted in Figure 32. The elements described herein are then discussed in detail in the upcoming Sections.

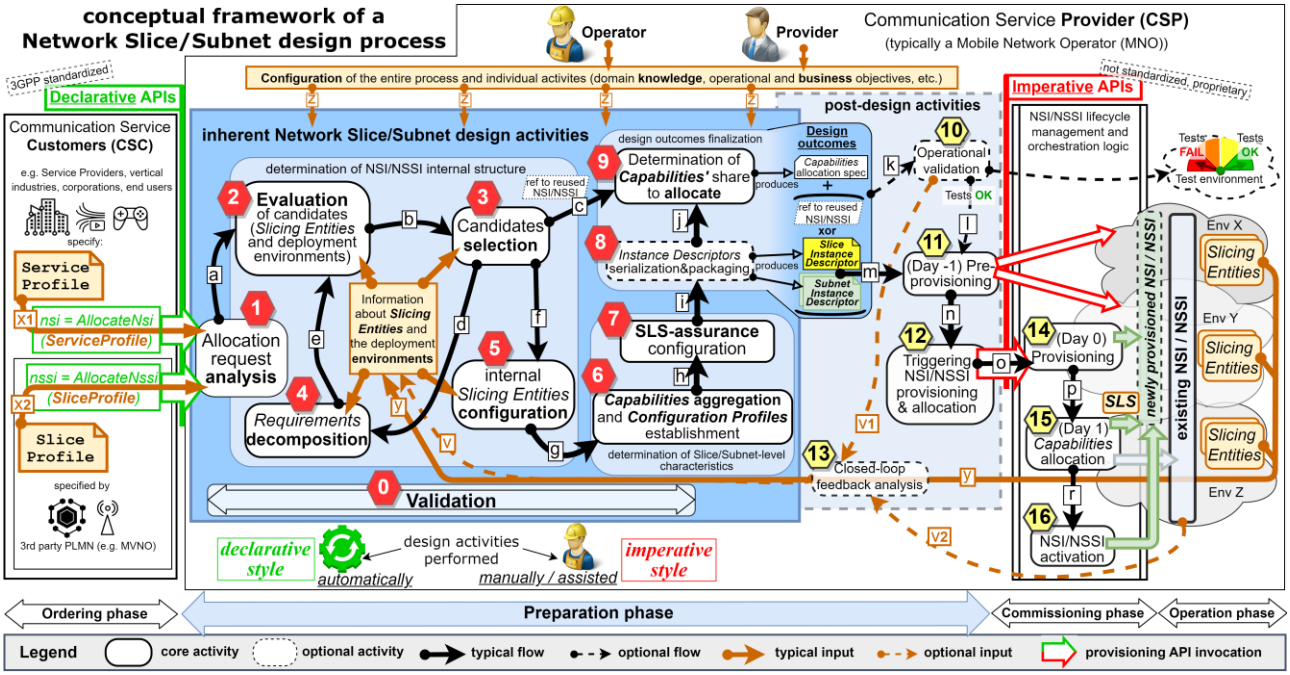


Figure 32. Schematic overview of the conceptual framework of a Network Slice/Subnet design process, distinguishing the inputs to the process (brown arrows), the inherent and post-design activities (ovals marked with numbered badges), its flows (black arrows), as well as the provisioning APIs which are either invoked before (*declarative* operations marked with green) or after conducting the design process (*imperative* operations marked with red).

Designing a Network Slice/Subnet is a complex process composed of multiple, coarse-grained **activities** represented graphically by rectangular ovals captioned with numbered badges. The core activities are denoted by ovals with solid borders, and the optional activities are distinguished by ovals with dashed borders. The activities marked by red badges with numbers 1-9 are considered **inherent** activities of the design process, as they are essential to produce the design outcomes. The **post-design** activities marked by yellow badges with numbers 10-16 act on the produced design outcomes and, as such, are considered outside of the design process itself. Nevertheless, their inclusion in the overall allocation process (at least the core ones) is essential to achieve end-to-end pipeline that results in the delivery of a ready-made NSI/NSSI that is allocated to fulfill the Customer's SLS *Requirements*. The arrows captioned with letters denote the

inputs to the design process (brown) and the **flows** of the process (black). The typical inputs and flows are represented by solid arrows, whereas the optional ones are marked with dashed arrows.

4.2.1. Inputs to the design process

As discussed in Subsection 2.3.2, the design process is triggered at the moment of obtaining the Customer's NSI/NSSI allocation request containing `ServiceProfile`/`SliceProfile` as schematically depicted in the left part of Figure 32 by inputs *"x1"/"x2"*. While the SLS *Requirements attributes* and their values contained therein serve as the primary input of the design process, there are also other pieces of information that need to feed it. These inputs, denoted with letters *"y"*, *"z"* and *"v"*, originate in the subsystems of the utilized Orchestrator, the deployment environments, as well as the human Operator and the Provider.

The basic **operational input** is the information about *Slicing Entities* possible to be utilized in the design process as well as the operational information about the deployment environments. To obtain such information, the design process needs to have access to the operational inventories containing the information about *Runtime* and *Deployment-time Slicing Entities*, as well as to the APIs allowing for obtaining their current and, possibly, historical/predicted state in terms of their operational status, configuration, monitored performance and faults (i.e. the CM/PM/FM data). The design process should also have access to the repositories storing the *Design-time Slicing Entities*: both the not-yet-on-boarded and the already on-boarded ones. In a similar manner, the design process should be allowed to utilize the APIs of the systems managing the deployment environments (e.g. of the VIM, CISM or WIM mentioned in Subsection 2.2.4) in order to obtain the current and, potentially, historical/predicted CM/PM/FM values within these environments. The totality of the operational input obtained from the aforementioned systems is represented by input *"y"* in Figure 32.

Prior execution, the design process needs to be appropriately **configured** by the human Operator and/or Provider, which is schematically indicated by inputs *"z"* in the upper part of Figure 32. The human Operator's essential responsibility is to deliver various pieces of **expert domain and operational knowledge**, including the specification of supported input SLS *Requirements attributes* and ranges of their values, *Capabilities attributes* exposed by the *Slicing Entities* as well as methods of their processing, specification of NSTs/NSSTs that define the abstract internal structure of the NSIs/NSSIs to be designed, as well as the operational policies that will adjust the whole process (or its individual activities) to the operational objectives to be achieved. In turn, the Provider should specify business policies that will influence the course of the design process to optimize the Provider's business objectives. Lastly, **closed-loop feedback** from the test and/or deployment environments, denoted by inputs *"v1"* and *"v2"* respectively, may feed the design process in advanced implementations, which is further discussed in the upcoming Subsection 4.6.2.

Once all the inputs have been described, the activities comprising the Network Slice/Subnet design process can be introduced.

4.2.2. Walkthrough of the inherent design activities

The **inherent** design activities begin with the **(1)** ***allocation request analysis***, aiming at its validation and determination of its handling policy in the further course of the design process, according to the Provider's and Operator's objectives. Then, the **(2)** ***evaluation*** of *Requirements* fulfillment by the considered *Slicing Entities* and the target deployment environments is performed, followed by the **(3)** ***selection*** of the most suitable ones to be utilized in the designed NSI/NSSI, by taking into account the determined policies. As detailed in the upcoming Subsection 4.3.2, the initial *evaluation* and *selection* may result in the choice of a *fully concretized Slicing Entity* representing a Slice/Subnet as a whole (e.g. an already running NSI/NSSI or a Slice/Subnet *Instance Descriptor*), in which case further design activities may be largely omitted (flow *"c"*). However, when a *not fully concretized Slicing Entity* is selected (e.g. an *Abstract Template* or a *Semi-concretized Descriptor*), the design process continues with **(4)** ***Requirements decomposition***, based on which the further, potentially recursive[119], *evaluation* and *selection* of internal, *lower-level Slicing Entities* and their target deployment environments is performed. Once all the internal *Slicing Entities* of the designed NSI/NSSI

[119] Recursive execution of the *evaluation*, *selection* and *Requirements decomposition* activities is performed in case of designing nested NSSIs.

have been selected, their **(5)** ***configuration*** need to be determined, in order to assure that that all of their settings have concretized values, which is an important prerequisite for their unattended instantiation and/or (re)configuration to fulfill the SLS *Requirements* in the designed NSI/NSSI.

While the abovementioned activities 2-5 are mainly concentrated on the individual, internal *Slicing Entities* building up the designed NSI/NSSI[120], activities 6 and 7 aim at defining its characteristics perceived as a self-contained, encapsulated unit[121]. This is achieved by performing the **(6)** ***Capabilities aggregation and Configuration Profiles establishment*** of the designed NSI/NSSI, followed by determining its **(7)** ***SLS-assurance configuration***. Over the duration of the design process, **(0)** ***validation*** shall be continuously performed, verifying the correctness of the partial results produced in the individual design activities.

Once all the already enumerated activities are successfully accomplished, *validation* should indicate design completeness and correctness - only then, the potential **(8)** ***serialization and packaging*** of *Instance Descriptors* to the representation accepted by the target Orchestrator may be performed. Finally, the activity that finishes the inherent part of the design process is the **(9)** ***determination of Capabilities' share to be allocated*** in the reused or newly designed NSI/NSSI – this share will be devoted for supporting the fulfillment of `ServiceProfile`/`SliceProfile` SLS *Requirements* during NSI/NSSI operation. As a result of the inherent design activities, the **design outcomes are produced** which, depending on the decisions made during the design process flow, are either *Instance Descriptors* of the newly designed NSI/NSSI or a reference to an existing one to be reused, as well as the specification of *Capabilities* allocation in the *Slicing Entities* comprising the NSI/NSSI for the fulfillment of the input `ServiceProfile`/`SliceProfile`.

Detailed description of the logic and automation methods of the inherent design activities is provided in Sections 4.3, 4.4 and 4.5.

4.2.3. Walkthrough of the post-design activities

The **post-design** activities typically include the **(11)** ***pre-provisioning*** (Day -1) procedures, including on-boarding of the created *Instance Descriptors* in the Orchestrator's subsystems, pre-configuration of the deployment environments and potential reservation of their resources. Optionally, the operational usage of the design outcomes may be preceded by their **(10)** ***operational validation*** in a test environment, in order to verify that the *Runtime Slicing Entities* spawned and/or (re-)configured based on the design outcomes indeed allow for satisfying the `ServiceProfile`/`SliceProfile` SLS *Requirements* stated as input to the design process. The test results may be used to approve (flow "*I*") or invalidate (flow "*v1*") the produced design outcomes for operational usage. Both the test results as well as the operational data obtained from the deployment environments (*"v2"*) may feed the advanced, optional **(13)** ***closed-loop feedback analysis*** activity, which aims at improving the design decisions in the subsequent execution of the design process. Once the target systems are prepared, the design outcomes are utilized for **(12)** ***triggering the NSI/NSSI provisioning*** (Day 0) ***and allocation*** (Day 1) procedures in the Orchestrator, marking the end of the preparation phase procedures in the end-to-end allocation process. Selected aspects of post-design activities and their automation are described in more detail in Section 4.6.

Triggering the NSI/NSSI provisioning and allocation procedures moves the end-to-end allocation process to the commissioning and operation phases of the NSI's/NSSI's lifecycle. The procedures executed in these phases depend on the outcomes of the design process. If a new NSI/NSSI has been designed, the information contained in its *Instance Descriptors* is utilized for performing the **(14)** ***(Day 0) provisioning*** operations that spawn and initially configure the utilized *Slicing Entities*, as well as populate the Orchestrator's operational inventories with appropriately configured management objects. Afterwards, the already existing NSI/NSSI and its internals are subjects of the **(15)** ***(Day 1) allocation*** procedures that result in, among others, the assignment of the decided share of their *Capabilities* to support the input `ServiceProfile`/`SliceProfile`. After **(16)** ***NSI/NSSI activation*** for serving the traffic of the supported CSI, the end-to-end allocation process is finished.

[120] These activities determine concretized values for the information items comprising the design outcomes discussed in Subsection 3.3.2.
[121] These activities determine concretized values for the information items comprising the design outcomes discussed in Subsection 3.3.3.

Once a high-level walkthrough of the framework has been provided, detailed elaboration of the most essential inherent and post-design activities as well as their automation methods is provided in the following Sections.

4.3. Logic and automation of activities determining NSI/NSSI internal structure

In order to devise method of automating the particular design activities, they are analyzed individually to specify their operational goals, processed data and the expected results. By gaining such an understanding, reasoning about their automation methods is discussed, leveraging the modeling constructs proposed in Chapter 3. In this Section, the analysis focuses on a crucial group of inherent design activities that aim at determining the internal structure of an NSI/NSSI capable of fulfilling the Customer's SLS *Requirements*.

4.3.1. Allocation request analysis

The first activity involved in the design process is the **(1) *allocation request analysis***. Among its basic responsibilities, the SLS *attributes* in the input `ServiceProfile`/`SliceProfile` data structure passed in the `allocateNsi`/`allocateNssi` invocation need to be **validated** syntactically and semantically before its further processing. Here it shall be reminded that almost all of the standardized SLS *attributes* in `ServiceProfile`/`SliceProfile` are defined as optional in the 5G NRM, as discussed in Subsection 2.3.3. Moreover, the request may also include vendor-specific SLS *attributes*. Hence, validation should confirm that the allocation request only contains *attributes* supported by the Orchestrator and their values are within the supported ranges[122]. **Automated validation** can be performed by confronting the input SLS *attributes* and their values against the configurations supplied to the design process provided by the Operator (see Subsection 4.2.1).

After successful validation, the **policies of handling** the allocation request shall be determined, to be utilized in the further course of the design process (flow *"a"*). Such policies may depend, for example, on the business status of the Customer requesting NSI/NSSI allocation, on the presence or value of the particular SLS *Requirements*, current state of the administered environment, etc. The applied policies influence or enforce particular decisions in the upcoming design activities, steer the flow of the design process and, as a result, affect the produced outcomes. Several examples of such policies have been already provided at the earlier stages of this dissertation, e.g. preferred reuse of the already existing *Runtime Slicing Entities*, utilization of a particular scoring vector during *evaluations*, etc. The policies may also control the admission of allocating NSI/NSSI of a particular type (SST[36]) by a given Customer, in a particular geographical area or impose limits on resources consumption. **Automated determination of handling policies** can be achieved based on the business and operational knowledge delivered by the Provider and the Operator, leveraging mechanisms proposed in Subsection 3.6, e.g. simple configuration files or reasoning rules evaluated by a policy engine.

4.3.2. Initial evaluation of candidates (representing Slice/Subnet as a whole)

In terms of high-level semantics, the **(2) *evaluation*** activity is about **assessing *Requirements* fulfillment** by the ***candidates*** considered at the particular stage of the design process, with the aim of providing their differentiated rating. *Evaluation* is a generic design activity that recurs at multiple stages of the design process, resulting in its execution in different contexts. More specifically, the *candidates* being evaluated at various stages of the design process can be: *Slicing Entities* representing Slice/Subnet as a whole, lower-level *Slicing Entities* representing internals, as well as the environments considered for the deployment of the *lowest-level Slicing Entities*. In order to devise appropriate methods of its automation, a thorough understanding of the desired operation in this activity needs to be gained in all such contexts.

To this end, the current Subsection is focused on analyzing the **initial** execution of the *evaluation* activity, when the *Requirements* being considered are the `ServiceProfile`/`SliceProfile` data structures that represent high-level SLS *Requirement*s against the desired NSI/NSSI perceived as an end-to-end, single encapsulated entity. Due to this fact, *Requirements* in such original form can only be utilized for the

[122] While in the typical cases, requests containing unsupported attributes or ranges of values should be denied, an additional, manual or semi-automated verification may be performed before a definite decision is made about handling such a request (e.g. demanding requests discussed in Subsection 4.7.3).

evaluation of *Slicing Entities* that represent a Slice/Subnet as a whole[123]. This activity is revisited in the upcoming Subsection 4.3.5 in the context of the *lower-level Slicing Entities* and the environments of their deployment, because *evaluation* of such *candidates* requires prior execution of the (4) *Requirements decomposition* activity that is described in Subsection 4.3.4.

At the technical level, the *evaluation* activity is about confronting the values of *Requirements attributes* with the values of the *Capabilities attributes* exposed in the *Configuration Profiles* of the considered *candidates*. *Evaluation* of a single *Requirement-Capability* pair is performed by (a) ***comparing*** their values, (b) **deciding** if these values can be treated as a ***match***, and, potentially, (c) computing a ***score*** expressing the level of *Requirement* fulfillment[124]. Based on the results of all *Requirement-Capability* pairs evaluation, it needs to be determined whether the entire set of *Requirements* can be considered fulfilled by the *candidate*, by computing its ***aggregated score*** - a single expression of how well the evaluated *candidate* fulfills the entire set of *Requirements*. Undoubtedly, performing the *evaluation* activity requires expert domain and operational knowledge about each attribute contained in the posed *Requirements* and in the *Capabilities* of the considered *candidates*. In the paragraphs below, several concerns regarding the attributes' *comparison, matching* and *scoring* methods are elaborated, taking into account their **data type** and **operational semantics**.

Comparing and deciding if a numerical or boolean *Capability* value *matches* (fulfills) the *Requirement* is in most cases straightforward. However, taking into account the interpretation of *Capabilities*' values exposed by *Slicing Entities* of different concretization and abstraction levels (discussed in Subsection 3.1.4), the *comparison* methods must support numerical *Capabilities* expressed as ranges of continues or discrete values. This is the case, for example, when the evaluated *Slicing Entity* is an *Abstract Template* or a *Semi-concretized Descriptor*, which, due to their not full concretization, may declare estimated ranges of *Capabilities* values in their *Configuration Profiles*, or the considered *Configuration Profile* enables *Slicing Entity*'s autonomous adaptation features, allowing it to offer its *Capabilities* in a range of values. To support such cases, utilization of more sophisticated *comparison* methods may be required, for example based on the Łukasiewicz [159] or fuzzy logic [160]. Assessing if such values can be treated as *matching* must be possible based on the potentially non-binary *comparison* results, which requires configuring the ranges or discrete values mapping to particular *matching* decision. For example, it must be decided if the guaranteed `dLLatency` *Capability* value of an *Abstract Template* expressed by a range (e.g. 10-20ms) is a *match* for a *Requirement* value specified as a concretized, threshold value (e.g. 15ms).

For the non-numerical attributes (e.g. complex data structures like sets, mappings, etc.), specialized *comparison* methods need to be implemented, tightly bound to the semantics of a particular *Requirement*/*Capability* attribute. For example, the `coverageAreaTAList` *Requirement* present in the `SliceProfile`, where a specific set of TAI[61] codes is required, will be compared with a set of geographical codes declared as the *Capability* value of a *candidate*. Moreover, dedicated logic must be implemented to reflect the Operator's policy if a superset of the covered geographical codes shall be treated as a *match*, or an exact coverage is required.

The *score* computed for the *evaluated* candidate might be expressed, for example, as a percentage or an enumerated value (e.g. "low", "medium", "high", etc.). It may also be supplemented with *detailed evaluation results* of comparison performed on the level of each *Requirement-Capability* pair, potentially containing their historical/predicted trends or other information that provides more insight into the computed *score* (e.g. averaged and peak values of the evaluated performance metrics, etc.). In such a case, even if the evaluated overall *score* would indicate advantage of one *candidate* over the others, the *detailed evaluation results* may influence the decisions made in the *selection* activity to optimize a particular aspect of the designed NSI/NSSI (discussed in the next Subsection 4.3.3).

[123] In a general case, *Slicing Entities* representing Slice/Subnet as a whole may include: existing NSIs/NSSIs, reserved-but-not-yet-instantiated NSIs/NSSIs, Descriptors of a Slice/Subnet (in the form of *fully concretized Instance Descriptors* or *Semi-concretized Descriptors*) or *Abstract Templates* of a Slice/Subnet. See Subsection 3.1.1 for a detailed elaboration of the *Slicing Entities* taxonomy.

[124] The computation of *score* may be omitted if the binary evaluation results (fulfills / not fulfills) are considered enough in the implemented design process. However, it must be realized that the lack of a *score* may not allow for a differentiated assessment of the evaluated *candidates* into those satisfying the *Requirements* in a better or worse manner.

A conceptual model that is capable of conveying the knowledge necessary for performing *evaluation* of a given *attribute* has been proposed in Subsection 3.4. By leveraging it, the *comparison*, *matching* and *scoring* methods for all supported attributes can be captured in the instances of such model. Then, the realization of an **automated *evaluation* activity** is merely about providing implementation of these methods and their execution for all *Requirement-Capability* pairs of the considered *candidates*.

4.3.3. Selection of candidates

Based on the *evaluation* results, potentially multiple *candidates* are considered for (3) ***selection*** in the further course of the design process, following flow *"b"*. If there are indeed multiple *candidates*, then a single, most suitable *Slicing Entity* and its specific *Configuration Profile* needs to be *selected*, according to the criteria established by the Operator or the Provider. Here it shall be underlined that it is the *selection* activity that allows for addressing the second aspect of the design process definition provided in Subsection 3.2.1, namely optimization of the designed NSI/NSSI according to the Provider's and/or Operator's objectives. For example, the Operator may aim at selecting a *Configuration Profile* that offers the best tradeoff between the level of *Requirements'* fulfillment and the anticipated *Slicing Entity*'s footprint. The reuse of the already existing NSIs/NSSIs in order to optimize the utilization of resources may also be promoted by the policies determined in activity (1). While typically the *candidate* will be selected from the ones that are evaluated as fully satisfying the posed *Requirements*, the Provider's policy may be to accept degraded performance or overutilization of resources by enabling overbooking [161]. Such decision makes *selection* of *candidates* that do not necessarily meet fully all stated *Requirements* possible. The *selection* policies applied by the Provider may also take into account the business status of the Customer requesting Slice/Subnet allocation (e.g. VIP, strategical, regular, low priority), the current situation in the deployment environments (e.g. current load, usage costs, scheduled maintenance, etc.), or any other business or technically oriented factors important for the Provider/Operator to be optimized.

Automation of the *selection* activity may be achieved with simple, easy to implement solutions based solely on the *evaluation results*, or may take into account a broader spectrum of factors influencing the final *selection* decision, which requires leveraging more sophisticated methods and tools. In a basic strategy, the *selection* activity simply picks any of the fulfilling *candidates*. An obvious extension of the simple strategy is not only to choose from the *candidates* evaluated in a binary manner, but to rely on their aggregated *score*, expressing their level of *Requirements'* fulfillment. Such a strategy allows for the selection from *candidates* based on their differentiated assessment, distinguishing *candidates* fulfilling the *Requirements* in a better or worse way. This approach allows for the application of dedicated methods for *selecting* the final *candidate*, in order to optimize the chosen operational aspects of the designed Slice/Subnet. For example: by sorting the *evaluation results* by the *candidates*' *aggregated score* (or *scores* assigned to the particular *Requirement-Capability* pairs), the Operator may configure one of the two strategies: either *select* the candidate which fulfills the *Requirements* with the highest or the lowest *score*. The first strategy allows for maximizing the overall level of *Requirements* fulfillment by the designed NSI/NSSI. The second one results in designing an NSI/NSSI from *candidates* fulfilling the *Requirements* to a minimum extent, which may contribute to more effective resources utilization.

In more advanced variants, automated *selection* should allow flexible optimization of a variety of technical or business aspects of the managed Telco infrastructure, according to the Operator's and/or Provider's policies. Several approaches of increasing sophistication level may be considered. A simple approach is to implement a scoring mechanism (similarly as in the case of the *evaluation* activity), in which the *evaluation results* serve just as one of many potential factors. Preferably, multiple variants of the scoring vectors specifying the influential aspects and their configured weights should be predefined to be applied in particular circumstances according to the Provider's/Operator's preferences. In more sophisticated approaches, the *scores* computed in a particular execution of the design process may be utilized to keep track of the *candidates*' **rating**. The rating information may also be computed as a result of the (13) *closed-loop feedback analysis* (discussed in the upcoming Subsection 4.6.2), constituting additional criteria in the subsequent executions of the design process, enhancing the *selection* decisions with a historical and operational outlook.

Policies evaluated by a policy engine deliver powerful tooling that allows the expression of the *selection* criteria in a flexible way by defining a set of reasoning rules determining the selection method in various conditions. In particular, policies allow not only for the application of different scoring vectors in particular circumstances (e.g. depending on the business status of the Customer requesting NSI/NSSI allocation) but also for fixing certain selection choices, which would be difficult to express by manipulating the scoring weights, e.g. a fixed choice of particular *Slicing Entities* vendor in certain circumstances. For example, for Customers with a "regular" status, a particular scoring vector is utilized, which assigns higher weights to the already existing *Runtime Slicing Entities* that fulfill the *Requirements* to the minimum extent, have the smallest footprint and do not utilize any costly hardware acceleration extensions. However, for the Customers with a "VIP" or "strategical" status, the scoring vector may prioritize utilization of *Slicing Entities* offering the highest degree of *Requirements* fulfillment, utilization of hardware acceleration, and it may also promote to spawn new instances of *Slicing Entities* instead of reusing the already existing, sharable *Runtime Slicing Entities*. A variety of other *selection* strategies can be expressed according to the Provider's/Operator's expert knowledge and operational/business objectives by defining customized reasoning rules.

After the execution of the *selection* activity, the design process may follow different flows depending on the concretization level of the selected *candidate*. The *selection* may result in the choice of a *fully concretized Slicing Entity*, i.e. an *Instance Descriptor* of a Slice/Subnet, a reserved or already existing NSI/NSSI to be reused. Such *Slicing Entity* may be utilized as is**,** with its currently exhibited *Capabilities* values that stem from its current *Configuration Profile* – in such a case a large part of the design activities may be skipped (flow *"c"* in Figure 32) because the structure of the selected candidate is already fixed. Alternatively, it may be reconfigured with another *Configuration Profile* to exhibit *Capabilities*' values better adjusted to the input SLS *Requirements*[125]. If, on the other hand, a *not fully concretized Slicing Entity* has been (3) *selected* or the structure of the *fully concretized Slicing Entity* shall be adjusted, then the design process follows flow *"d"* to *decompose* the Customer's input SLS *Requirements*.

4.3.4. Requirements decomposition

The aim of the **(4)** ***Requirements decomposition*** activity is to transform the input, higher-level *Requirements* into a set of their lower-level counterparts. Such a need occurs when an *Abstract Template* or a *Semi-concretized Descriptor* representing Slice/Subnet is selected to be utilized for designing the NSI/NSSI as a whole, or its nested NSSI(s). In such a case, their **contextualization** needs to be performed, in order to map the not explicitly identified internals contained in these *Design-time Slicing Entities* with precisely identifiable *lower-level Slicing Entities* available in the administered environments. To accomplish this mapping, *evaluation* and *selection* of such candidates must be performed, which is described in the following Subsection 4.3.5.

However, the high-level SLS *Requirements* in the initial form of a `ServiceProfile`/`SliceProfile` obtained from the Customer in the `allocateNsi`/`allocateNssi` request are not suitable to *evaluate* the *lower-level Slicing Entities*, as these *Requirements* pertain to a Slice/Subnet perceived as an end-to-end unit. Hence, these high-level *Requirements* need to be *decomposed* to the lower-level ones which, with their semantics and abstraction level, will be possible to be posed against the *Capabilities* exposed by the *lower-level Slicing Entities*. Moreover, it needs to be noted that the *Requirements decomposition* activity shall not only produce the lower-level *SLS-satisfaction Requirements*, but also the *functional* and *composability Requirements*. This is because *evaluation* of the *lowest-level Slicing Entities* against such *Requirements* is indispensable to assure that the considered *candidates* deliver the expected behavior and fit the topology of internals assumed by the utilized *Abstract Template* or *Semi-concretized Descriptor* (see Subsection 3.1.2).

Requirements decomposition is a challenging activity that requires expert operational and domain knowledge for appropriate understanding and interpretation of the higher and lower-level *Requirements*, as well as technical knowledge of how the particular higher-level *Requirements* can be satisfied with the combination of *Capabilities* offered by the particular lower-level *Slicing Entities* and their concretized *Configuration*

[125] If such decision is made, it shall be assured that there are enough resources to perform the reconfiguration. Moreover, in the case of a sharable *Slicing Entity,* it must be verified that the reconfiguration will not negatively influence the satisfaction of SLS *Requirements* already assigned to it.

Profiles. It is also among the most critical activities of the design process, as it drives its further course by influencing the decisions possible to be made at its later stages. It is because the decomposed, lower-level *Requirements* serve as input for the (2) *evaluation* and (3) *selection* of *lower-level Slicing Entities*, which have significant impact on the finalized design outcomes.

Capturing the knowledge necessary for performing this activity is possible in the instances of the *Requirement Decomposition Rules* (RDR) conceptual model proposed in Subsection 3.5.1. Therein, three generic transformations comprising an RDR (*translation*, *assignment* and *value distribution*) have been explained and multiple examples of their semantics have been provided. Basing on this modeling construct, **automation of the *Requirements decomposition* activity** is merely about implementing the particular methods assumed in the instances of the RDRs and combining their execution into a *translation→assignment→value distribution* pipeline executed according to the configuration contained in the RDR instances. Such realization allows for obtaining the desired results in the form of lower-level *Requirements attributes* with decided values that can be used for the *evaluation* of the *lower-level Slicing Entities*, as described in the next Subsection.

4.3.5. Subsequent evaluation and selection of candidates (lower-level Slicing Entities, deployment environments)

Although the (2) *evaluation* activity has already been described in Subsection 4.3.2, in the context of its initial execution, in the current Subsection this activity is analyzed in its subsequent executions at the later stages of the design process. The need for its repetitive execution occurs when an *Abstract Template* or a *Semi-concretized Descriptor* representing a Slice/Subnet is selected in the initial or in the recursive executions of the *selection* activity. In such a case, after obtaining the lower-level *Requirements* as a result of the (4) *Requirements decomposition* activity, its **contextualization** needs to be performed to determine the *lower-level Slicing Entities* that will be utilized as internals of the designed NSI/NSSI. More specifically, if a Slice/Subnet *Abstract Template* has been *selected*, its abstractly defined structure of internals must be **mapped** to the *lower-level Slicing Entities* that meet the decomposed *Requirements*. Such mapping may also be needed to contextualize the selected *Semi-concretized Descriptor* of a Slice/Subnet, which may leave references to some of its internals unspecified, or may allow/enforce the reuse of the already existing *Runtime Slicing Entities* in its assumed structure.

What distinguishes the *evaluation* of *lower-level Slicing Entities* is that not only the decomposed, lower-level *SLS-satisfaction Requirements* are assessed, but also the *functional* and *composability Requirements* (produced as a result of the *Requirements decomposition* activity) need to be *evaluated* against the *Capabilities* of the considered *candidates*. Moreover, e*valuation* of the *Slicing Entities* themselves is only one side of the coin. When considering *Design-time* descriptions of the *lowest-level Slicing Entities*, also the **deployment environments** for their potential provisioning need to be evaluated. For example, utilization of a given MF Descriptor or its particular *Configuration Profile* may be evaluated as not fulfilling the *Requirements*, even if the evaluated *Slicing Entity* itself fulfills them, when its infrastructural demands cannot be met in none of the considered deployment environments, e.g. the hardware acceleration extensions are not available. Furthermore, the remaining capacity of the considered deployment environments needs to be checked as well, in order to verify that allocating the share of resources assumed by the particular *Configuration Profiles* is still feasible. In other words, only by combining both the results of evaluating the *lowest-level Design-time Slicing Entities* and the environments for their potential provisioning provides the total evaluation information based on which the suitability of utilizing it in the designed NSI/NSSI can be assessed.

In order to perform **automated *evaluation* of deployment environments**, similar methods as in the case of *Slicing Entities evaluation* (i.e. *comparing*, *matching*, *scoring*) may be utilized. Such approach requires modeling the deployment environment characteristics to be exposed as their *Capabilities*[126], which should correspond to the *Requirements* posed against them. Such *Capabilities attributes* and their values may be

[126] *Capabilities* of deployment environments should not be confused with the `NetworkSliceSubnetProviderCapabilities` model, which was added in 5G NRM in version 17.7.0 (2022-06) [204]. This model enforces the exposition of *SLS-satisfaction Capabilities* by the entire NSSMF block, not the particular deployment environments it manages behind the scenes, which are considered for deployment of the *lowest-level Slicing Entities*.

statically defined (e.g. the geolocation of the PoP in which deployment environment resides, or a database containing the coverage area of the available base stations) or determined by aggregating information from the PM/FM interfaces of the Orchestrator's subsystems[127]. Besides the aforementioned "raw" information sources, the Data Analytics Functions (DAF) [162], both at the Network Functions level (NWDAF) and management level (MDAF), or the Self-Organizing Networks (SON) functionalities of the target technological stack shall all be considered as valuable data sources, based on which the *Capabilities* of the deployment environments can be determined and exposed.

After accomplishing the *evaluation* activity, the *evaluation results* are utilized as the main input for the (3) ***selection*** activity, which is performed as already described in Subsection 4.3.3. As a result, a single *Slicing Entity* and its *Configuration Profile* for each of the mapped internals in the utilized *Abstract Template* or *Semi-concretized Descriptor* shall be determined. If among the selected *Slicing Entities* there are again *Abstract Templates* or *Semi-concretized Descriptors* that specify the structure of nested Subnets, their contextualization is performed by recursively following flow *"d"* towards further (4) *Requirements decomposition*, (2) *evaluation* and (3) *selection* of their internals. These activities are executed repetitively until all internals are contextualized. Then, the design process proceeds with flow *"f"* towards the (5) *configuration* activity described below.

4.3.6. Configuration of internals

After the structure of the designed NSI/NSSI has been established by determining precise identities of its internal, non-abstract *Slicing Entities*, the design process moves towards their **(5) *configuration***. The aim of this activity is to determine all the missing information that will make the produced Slice/Subnet *Instance Descriptors fully concretized*. Full concretization of the design outcomes is essential for automating the end-to-end allocation processes, as it assures the readiness of the design outcomes for unattended utilization by the Orchestrator.

Configuring internal *Slicing Entities* to make them ready for operation in the designed NSI/NSSI requires detailed, highly-specific knowledge about them. Hence, automating the (5) *configuration* activity in a generic case is difficult to achieve without imposing strong assumptions on the processed data. Indeed, the concept of *Slicing Entity Configuration Profiles* has been proposed by the author mainly to ease the configuration burden during the design process. This modeling construct puts the obligation of preparing the particular *Slicing Entities* for automated processing on the *Slicing Entity* Vendor or the Operator, who are obliged to predefine coherent configuration values for the "impactful" attributes and provide credibly tested *Capabilities*' values that can be expected from such *Slicing Entities* with a given set of configurations applied. Hence, the author claims that in the context of an automated design process, enforcement of *Configuration Profiles* preparation is a simplifying, but at the same time, mandatory prerequisite for its organization. Lack of such a prerequisite would entail the need of "trying" different combinations of configuration values for a potentially large range of "impactful" knobs, and "guessing" what level of performance can be expected from such a *Slicing Entity* with the given configurations applied.

If the aforementioned assumption is met, values for a large fraction of configuration attributes, at least for the "impactful" ones, are automatically determined by choosing the appropriate *Slicing Entity Configuration Profile* in the (3) *selection* activity. Thus, **automation of the (5) *configuration* activity** is confined to the determination of values for the remaining configuration attributes that are not covered in the *Configuration Profile*. These attributes are the ones classified as "dependent" and "not impactful & independent" (see Table 5, columns F and G and Figure 32). Typically, automated delivery of values for such attributes needs to be considered in a case-by-case manner. Column D in Table 5 states recommendations for approaching automated delivery of values for the standardized attributes. It must be noticed, though, that the majority of attributes to be configured in a given *Slicing Entity* are usually vendor-specific. As such, automating the delivery of their values may need to be handled by a dedicated, potentially highly-specific logic.

[127] These may be the generic 3GPP PM/FM interfaces [73] at the level of NSMF/NSSMF/NFMF blocks or NFVO/VNFM [89], storage/networking/ computing management interfaces of VIM [205], but also APIs exposed by the virtualization platforms like OpenStack, Kubernetes or public cloud provider.

In the context of the "dependent" configuration attributes (column F, Table 5), there is an important challenge of composing the internal, *lower-level Slicing Entities* to configure their endpoints (`EP_RP` of MFs, `EP_Transport` of Subnets) with information about the far-end *Slicing Entity* to which it shall be connected. In the case of MFs in the Core domain, this problem has been largely mitigated by the advent of SBA and its enforcement by the 3GPP standards (see Subsections 2.1.2 and 2.1.5). Thanks to SBA and the central role of NRF, the bindings between the MFs that expose an SBI endpoint among their *composability Capabilities* will be dynamically established in runtime. Nevertheless, for the MFs in the RAN domain and for the non-SBI-enabled MFs in the Core domain (e.g. UPF), their composition must be explicitly arranged, based on the endpoints exposed in their *composability Capabilities* and the topology of internal elements dictated by the utilized *Abstract Template* / *Semi-concretized Descriptor*. Thus, the concretized configuration of endpoints (e.g. IP addressing scheme for the N2, N3 endpoints between RAN and Core) must be provided to the peer *Slicing Entities*, to construct a well-functioning whole. Thanks to the *Requirements-Capabilities* evaluation and matching, automated configuration of the "dependent" configuration attributes can be achieved.

4.4. Logic and automation of activities determining Slice/Subnet-level characteristics

The design activities discussed in the previous Section have been concentrated on the *Slicing Entities* constituting the internal building blocks of the designed NSI/NSSI. Once the internals have been (3) *selected* and (5) *configured*, the design process moves with flow *"g"* towards a block of inherent design activities (6)-(7) that aim at determining consolidated characteristics of the designed NSI/NSSI perceived as a single, encapsulated unit. The essence of these activities and the proposed methods of their automation are described in the following Subsections.

4.4.1. Determining Slice/Subnet-level Configuration Profiles and aggregated Capabilities

The *Configuration Profiles* of the internal *Slicing Entities* that have been *selected* at the earlier stages of the design process formulate a distinguished "primary" *Configuration Profile* of the designed NSI/NSSI – as such, its automated determination is a consequence of the design decisions that were already made. However, the concretized *Capabilities* values to be exposed by the "primary" *Configuration Profile* need to be determined. For this purpose, the *Capabilities* of the internal *Slicing Entities*, which potentially exhibit lower-level semantics, need to be **(6)** ***aggregated*** to the ***Slice/Subnet-level Capabilities*** in each of the established *Configuration Profiles* (discussed in the upcoming paragraphs).

Aggregation of the *functional* and *composability Capabilities* is typically straightforward. The *functional Capabilities* are determined by re-exposing the *functional Capabilities* of internal *Slicing Entities*, potentially filtering out the ones which are irrelevant outside of the designed NSI/NSSI. In turn, the *composability Capabilities* comprise the endpoints (`EP_RP`, `EP_Transport`) of internal *Slicing Entities* that have not been linked to another *Slicing Entity* inside of the designed NSI/NSSI. The aggregation of the *SLS-satisfaction Capabilities*, however, is highly dependent on the semantics of the *Capabilities* exposed by the *lower-level Slicing Entities* and the higher-level *Capability* that shall be exposed as a result of their merging[128]. Hence, if **automation of this activity** is aimed to be achieved, specification of detailed knowledge about *Capabilities* to be aggregated needs to be captured and made available to be processed programmatically.

Determining the *aggregated Capabilities* is an operation that can be largely viewed as reversed to the (4) *Requirements decomposition* activity, described in Subsection 4.3.4. Indeed, when decomposing a single higher-level *Requirement*, its value is distributed among potentially multiple lower-level *Requirements*, whereas *Capabilities* aggregation is about merging values of potentially multiple lower-level *Capabilities* into a single value of a higher-level *Capability*. Thanks to this observation, an analogical approach for **automating the aggregation activity** can be utilized, by leveraging the generic model of a *Capability Aggregation Rule* (CAR) proposed in Subsection 3.5. Once CAR instances filled with the specification of the *source Capabilities*

[128] Several examples of *Capabilities aggregation* have been provided in Subsection 3.5.

and the *aggregation method* are defined for each high-level *Capability* to be exposed by the designed NSI/NSSI, they can be processed by an automated mechanism that simply implements the specified methods and executes them in a pipeline. As a result, a set of high-level *Capabilities attributes* with determined values can be automatically produced. Importantly, calculation of the *aggregated Capabilities* values should be performed for each of the *Configuration Profiles* that will be established in the designed NSI/NSSI, as discussed below.

Besides the "primary" *Configuration Profile*, **alternative *Configuration Profiles*** may be established in the designed NSI/NSSI, in order to allow their application in varying circumstances during its lifecycle. Here it needs to be pointed out that the *Requirements* values specified in the input `ServiceProfile`/`SliceProfile` represent the characteristics demanded from the NSI/NSSI operating under the contracted, maximum load, e.g. `maxNumberofUEs`, maximum tolerable `dLLatency`, `maxNumberofPDUSessions`, etc. However, during the operational lifecycle, it is unlikely that the deployed NSI/NSSI will require such a performance level all the time. Hence, additional *Configuration Profiles* may be established in order to enhance the operational dynamicity of the designed NSI/NSSI and its (self-)adjustment to the experienced conditions, optimizing the utilization of resources or strengthening the business relationship with the Customers. Generation of alternative *Configuration Profiles* has been extensively covered by the author in Section VI.F of [30] and is mentioned here only for the sake of completeness. However, for the sake of brevity, detailed elaboration of this aspect in this dissertation is skipped, as it is not crucial from the perspective of thesis verification and, as such, is not tackled in the experimental verification (see the PoC implementation assumptions described in Section 5.1).

After the accomplishment of the (6) *Capabilities aggregation and Configuration Profiles establishment* activity, satisfaction of the high-level input SLS *Requirements* by the *aggregated Capabilities* of the established *Configuration Profiles* (at least the "primary" one) should be verified as one of the (0) *validation* steps executed over the course of the design process - this can be achieved in an automated manner by reusing the generic (2) *evaluation* activity, executed with the determined set of *aggregated Capabilities* and the input SLS *Requirements*.

4.4.2. Determining Slice/Subnet-level SLS-assurance configuration

Once the Slice/Subnet *Configuration Profiles* have been established and their aggregated *Capabilities* values have been calculated, the design process moves with flow *"h"* towards the activity which aims at determining the **(7) *Slice/Subnet-level SLS-assurance configurations*** (mentioned in Subsection 3.3.3). The *Capabilities*' values of Slice/Subnet-level *Configuration Profiles* defined in the designed NSI/NSSI, the operationally monitored values of KPIs and performance measurements, values of the corresponding thresholds as well as the alarms of their crossing – all these information items and potentially other measures serve as input for defining **rules** of the **SLS-assurance policies**.

Among multiple possibilities to react to the degraded performance of the designed NSI/NSSI in operation, typically the SLS-assurance policies' rules should dictate to scale it up by changing its *Configuration Profile* to the one which offers *Capabilities*' values better adjusted to the currently experienced or predicted conditions (if such *Configuration Profiles* have been established). Similarly, during times of low utilization, the designed Slice/Subnet should scale down, by applying a *Configuration Profile* which consumes less resources or energy, but allows satisfying the Customer's SLS *Requirements* in the currently experienced or predicted load. To build such rules, the *aggregated Capabilities*' values of the Slice/Subnet-level *Configuration Profiles* and the alarms about crossing their thresholds should be taken into account, assigning a concretized behavior to be taken in the particular conditions.

Determination of appropriate SLS-assurance behavior in response to the monitored state of the designed NSI/NSSI is a challenging task, requiring advanced knowledge about the utilized internal *Slicing Entities*, their offered scaling and adaptation features as well as the SLS-assurance functionalities offered by the utilized Orchestrator. As such, configuration of the SLS-assurance policies is typically based on the experience gained through empirical studies and closed-loop feedback from running an NSI/NSSI (or its part) in the test and/or production environments, where the particular SLS-assurance strategies are tested in simulations based on

operational data gathered from the target runtime environments. **Automated establishment of SLS-assurance policies** may be attempted by their **generation** based on several indicators, which is covered by the author in Section VI.G of [30] and is mentioned in this thesis only for the sake of completeness. However, for the sake of brevity, detailed elaboration of this aspect in this dissertation is skipped, as it is not crucial from the perspective of thesis verification and, as such, is not tackled in the experimental verification (see the PoC assumptions described in Section 5.1).

After the accomplishment of the SLS-assurance configuration activity, all information items required for the production of a *fully concretized* Slice/Subnet *Instance Descriptor* have been determined, which enables the process to proceed with flow *"i"* towards a block of design activities that aim at finalizing the design outcomes, which are described in the following Section.

4.5. Logic and automation of activities finalizing the design outcomes

The design activities described until now aimed at selecting an already existing or defining the structure of a new NSI/NSSI. However, in order to finish the inherent part of the design process, several activities that finalize the produced outcomes must be accomplished, as described in the following Subsections.

4.5.1. Validations

Over the course of the design process, the intermediate results produced by each individual design activity shall be continuously **(0) *validated***. Besides verifying that the *aggregated Capabilities* of the newly designed Slice/Subnet meet the input `ServiceProfile`/`SliceProfile` *Requirements*, which has been mentioned in Subsection 4.4.1, there are other *validations* that shall be performed during the design process. Such *validations* include syntactic (i.e. all mandatory attributes required by the utilized Orchestrator are present, values are of assumed data types, etc.) and semantic (i.e. complex constraints checking, etc.) verifications, assuring that the design outcomes will be accepted as correct inputs for the Orchestrator and its subsystems. Other validation steps, namely the (10) *operational validation* and *pre-instantiation validation* (performed as a part of activity no. 13), are discussed as a part of the post-design activities in Subsection 4.6.1, as they may not be conducted during the inherent design activities.

Once all the already described design activities are finished, the final syntactic and semantic *validation* shall confirm that values for all information items required by the Orchestrator to perform the instantiation of a new NSI/NSSI have been determined. This means that the design reached the maturity level indicating its completeness, which allows the process to proceed with flow *"i"* to output the finalized form of *Instance Descriptors*, as discussed in the following Subsection.

4.5.2. Instance Descriptors serialization and packaging

In order to produce the finalized *Instance Descriptors* of the newly designed NSI/NSSI, all the information gathered over the course of the design process need to be represented according to the structure and format of the management models accepted by the target Orchestrator and its APIs. If **files** shall be the embodiment of the *Instance Descriptors*, then **serialization** of the data gathered during the design process must be performed. As already discussed, currently there are no standards defining the structure, semantics or format of any Network Slicing design-time models. Nevertheless, the currently observed common trend in the Telco ecosystem is towards serializing Descriptors into TOSCA-based dialects [163] (to express the topology of internals, their types, deployment arrangements and operational workflows), supplemented by YANG [91] based configurations and other files, including binary images, scripts, etc. It shall also be reminded that in the 3GPP-ETSI-compliant solutions, the "platform-layer" and "virtualized entity-layer" information items (identified in column E of Table 5 in Subsection 3.3.2) of the designed Subnets shall be serialized into the NFV-MANO NSDs according to the standardized TOSCA format [85] and their `CreateNsRequest`, `InstantiateNsRequest` contextualization data structures, which also need to be created for newly designed NSI/NSSI, as argued in Subsection 3.3.5.

As shown above, the overall design outcome describing all entities of the newly designed NSI/NSSI may be represented by **multiple files** or even multiple **packages** of files (e.g. CSAR [163] packages) containing

complete sets of information required for the instantiation and operation of each entity in the particular subsystem of the utilized Orchestrator. Additionally, creation of an "uber-package" may be considered (e.g. embodied as a variant of the CSAR package), encapsulating all aforementioned elements into a single file, which at the moment of operational usage is unpacked and its elements are disseminated to the corresponding subsystems. Other representations of *Instance Descriptors* are not precluded. For example, if the design process is tightly integrated with the Orchestrator, then they might be created as **in-memory objects**, without the need of serializing them. Regardless of the output form of the *Instance Descriptors*, the design process follows flow *"j"* towards the last of the inherent design activities, which is discussed in the following Subsection.

4.5.3. Determining the share of Capabilities to be allocated

Regardless of the decision to use an already existing *fully concretized Slicing Entity* representing NSI/NSSI as a whole (flow *"c"*) or to design a new NSI/NSSI (flow *"d"→"j"*), the final inherent design activity to be performed is the **(9) *determination of Capabilities' share to be allocated***. Although at the earlier stages of the design process it has been validated that the *Configuration Profile* of the determined NSI/NSSI (selected to be reused or the newly designed one) offers *Capabilities* satisfying the input `ServiceProfile`/`SliceProfile`, the actual share of *Capabilities* to be devoted in each of its comprising internal *Slicing Entity* for supporting them may be a subject of a deliberate design decision that is influenced by the Operator's/Provider's operational and/or business objectives. This activity is especially relevant for the *fully concretized Slicing Entity* representing NSI/NSSI that was selected to be reused, as its selection at the previous stage of the design process was decided based purely on the evaluation of its exposed high-level *Capabilities*, without precisely deciding how its internal *Slicing Entities* will contribute to the fulfillment of the high-level input SLS *Requirements*.

The simplest and the most straightforward approach to determine these shares in an **automated manner** is by utilizing the values produced in the (4) *Requirements decomposition* activity, which assigns the lower-level *Requirements* with decided values to the *Capabilities* of the internal *Slicing Entities*. Such values have been already produced in the case of the newly designed NSI/NSSI (see Subsection 4.3.4), but they can also be produced for the selected *fully concretized Slicing Entity* representing NSI/NSSI, by executing the (4) *Requirements decomposition* activity based on the RDRs contained in its corresponding design-time description. These values may be taken exactly as they are or, depending on the attributes' data type and operational semantics as well as on the determined policies, increased or decreased.

As an example, the decomposed *Requirement* against the gNB-DUs of the RAN Subnet may state the need for devoting 5% of the PRBs[8] in its managed Cells in order to satisfy the high-level *Requirements*. But, due to the VIP status of the Customer, which is taken into account in the Provider's policies, this value is increased by 20% in order to provide higher chances of SLS satisfaction, resulting in 6% of allocated PRBs. Conversely, if overbooking is deliberately enabled by the Provider, then the decision may be to decrease the value of some *Capabilities* to be allocated, e.g. lowered-down scheduling priority of the downlink/uplink traffic for Customers with a lower business status.

Once the share of *Capabilities* to be allocated in each *Slicing Entity* comprising the selected or newly designed NSI/NSSI has been determined, all the information items specified in the design outcomes definition (see Subsection 3.2.2) have been produced, which finishes the inherent part of the design process.

4.6. Logic and automation of post-design activities

In the NSI/NSSI allocation pipeline there are several activities that strongly relate to the design process, but are focused on utilizing its finalized outcomes and, as such, are considered outside of it. While the **post-design activities** (schematically depicted in Figure 32 with yellow badges) are not the main subject of this dissertation, for the sake of completeness of the presented framework, their most important aspects are briefly described in the Subsections below. Their comprehensive elaboration, including discussion of their detailed logic, automation opportunities and involved systems is provided in Sections V.L and VII.H of [30].

4.6.1. Pre-provisioning of design outcomes

Performing various kinds of (0) *validations* during the inherent design activities, although allowing for the verification of the correctness and completeness of the design outcomes, may be considered insufficient in Network Slicing, as it is the delivery of appropriate level of QoS satisfying the input SLS *Requirements* that plays the key role in this technology. Thus, before proceeding directly to operation, as indicated by flow *"m"*, the post-design process may proceed with flow *"k"* to perform **(10) *operational validation*** of the selected NSI/NSSI or of the produced *Instance Descriptors*, by executing a series of SLS-related performance and load tests during its dry run. Skipping the *operational validation*, by following flow *"m"*, can be considered in mature Network Slicing solutions, where the confidence in the credibility of characteristics exhibited by the *Slicing Entities* as well as the reliability and deterministic performance of the target deployment environments is high.

Once *operationally validated*, the particular parts of the created *Instance Descriptors* need to be **(11) *pre-provisioned*** in the Orchestrator's subsystems prior their operational usage. Due to the fact that the standardized structure of a Slice/Subnet is spread among the 3GPP and ETSI NFV-MANO orchestration domains, the static, not instance-specific parts of *Instance Descriptors* typically need to be on-boarded in the target subsystems prior their operational usage. For example, the VNF Packages backing the MF *Design-time Slicing Entities* which were not yet on-boarded in the target VNFMs or NFVOs shall be on-boarded according to the procedures described in [88] or [89], respectively. In turn, the NSDs created as a result of the design process must also be on-boarded in the NFVOs, according to the procedures described in [89].

Similar procedures, although not standardized, may need to be performed in the 3GPP orchestration blocks (NSMF, NSSMF) to on-board the created Slice/Subnet *Instance Descriptors*. Moreover, any other subsystems comprising the utilized Orchestrator may need to be pre-provisioned with the design-time information in order to prepare them for operating the *lowest-level Slicing Entities* utilized in the designed NSI/NSSI. For example, the EMS may need to be pre-provisioned with the YANG models, necessary to enable communication with the MFs, or the transport network manager(s) may need to be pre-provisioned with the configurations of the DCIs[103], etc. All such pre-provisioning activities shall be considered a part of the "preparation of the network environment" lifecycle step that is assumed in the NSI/NSSI preparation phase by the 3GPP standards (see Subsection 2.3.1).

Depending on the supported features and APIs offered by the utilized Orchestrator, as a part of the pre-provisioning activity, the information contained in the created *Instance Descriptors* may be used for resources **reservation**, by executing the procedures described in [31][129]. Reservation can be utilized for performing a deferred, scheduled instantiation of the newly designed NSI/NSSI. However, typically, the produced *Instance Descriptors* are used for immediate, **on-demand instantiation** of the newly designed NSI/NSSI, as described in the upcoming Subsection 4.6.3.

4.6.2. Closed-loop feedback

As an advanced post-design activity, the results of the executed (10) *operational validation* tests may be passed along input *"v1"* to be used for **(13) *closed-loop feedback analysis***, which aims at improving the design decisions in the current and next executions of the design process (e.g. improved rating of *candidates* in the (3) *selection* activity, as described in Subsection 4.3.3). In particular, the currently performed design process should take into account the correspondence between the *SLS-satisfaction Capabilities* declared by the considered *Slicing Entities* in the prior design process and the real, operationally collected KPIs and PM/FM metrics that measure the operational fulfillment of those declarations. Even if the *operational validation* is not being performed in the target solution, the primary source of feedback information may be based on the measurements collected from the operationally deployed NSIs/NSSIs (indicated by input *"v2"* in the lower-right part of Figure 32), which shall provide the most credible verification of the design outcomes.

[129] In the recent revisions of the 5G NRM (from version V17.7.0 [204]) and of the TS 28.531 [31] 3GPP standards, feasibility check as well as reservation models and procedures have been defined.

4.6.3. Triggering NSI/NSSI provisioning and allocation procedures

By accomplishing the (11) *pre-provisioning* activity, the design outcomes are ready to be utilized operationally, by following flow *"n"* to **(12)** ***trigger NSI/NSSI provisioning and allocation*** procedures in the Orchestrator. Here it needs to be reminded that the design outcome is a static piece of information which essentially "hibernates" all the decisions taken during the design process. Due to the dynamics of the target deployment environments (discussed further in the upcoming Subsection 4.7.4), Orchestrator should perform a pre-instantiation validation at the moment of accepting the request, in order to verify if, in the currently experienced conditions, it is feasible to perform the provisioning and allocation procedures according to decisions conveyed by the design outcomes.

Then, the end-to-end process moves with flow "o" towards execution of the **(14)** ***(Day 0) provisioning*** procedures in the Orchestrator's subsystems, which is only performed if a new NSI/NSSI has been designed. During provisioning, the relevant information of the *Configuration Profiles* established in the created *Instance Descriptors* is utilized to feed the requests for the instantiation and initial configuration of the internals comprising the NSI/NSSI in both the 3GPP Slicing and the underlying ETSI NFV-MANO layers. Based on the delivered design outcomes, these subsystems spawn and initially configure the utilized *Slicing Entities*, as well as populate the operational inventories with appropriately configured management objects.

After provisioning, the end-to-end process moves with flow *"p"* towards the **(15)** ***(Day 1) allocation*** procedures, where the newly created or reused NSI/NSSI as well as its internals are configured for supporting the input `ServiceProfile`/`SliceProfile` by assigning the decided share of their *Capabilities* for its fulfillment. Finally, the end-to-end allocation process follows flow *"r"* towards **(16)** ***NSI/NSSI activation*** with the `ServiceProfile`/`SliceProfile` assigned. This activity modifies the operational state of all *Slicing Entities* comprising the NSI/NSSI to the activated state, which admits the traffic of the supported CSI to be served by them. Successful execution of these procedures finishes the end-to-end allocation process.

4.7. End-to-end design process automation

After presenting the logic and automation opportunities of each design activity separately, this Section discusses important aspects of Network Slice/Subnet design process **end-to-end automation** that need to be taken into account in its realization. Firstly, Subsection 4.7.1 discusses various scopes of implementation of the proposed framework, its models and methods, identifying operational assumptions that simplify achieving end-to-end automation. Afterwards, in Subsection 4.7.2, a roadmap for gradual, backward-compatible rollout of end-to-end design automation is proposed. Then, in Subsection 4.7.3, the desired level of automation in the design process is discussed. Finally, Subsection 4.7.4 discusses several aspects that need to be taken into account in the technical realization of automated design pipelines according to the proposed framework, including the duration of the design process and its dynamicity.

4.7.1. Assumptions simplifying achieving end-to-end automation

Implementation and integration of all the models and mechanisms proposed in this thesis in their full extent may require significant development efforts, not only in the Orchestrator itself but also in its dependent subsystems. Thus, before attempting, the scope of the implemented solution should be confronted against the current features of the Orchestrator being augmented and the actual operational expectations to be satisfied. If permissible in particular circumstances, several **simplifying assumptions** presented below should be considered. By their application, end-to-end automation may be achieved with reduced scope of development, reduced implementation complexity of automation methods or relaxed requirements for the presence of specific data.

First, obvious simplification is to implement the simplest methods in each of the discussed design activities. When following such an approach, some of the advanced features (e.g. complex candidates' *selection* rules, enablement of overbooking, etc.) may not be available, but it allows for the inception of the basic structure for an automated system, which can already be tested and gradually supplemented with the most demanded features. Another simplification is not to implement all activities assumed by the framework from the very beginning. End-to-end design process automation in a limited operational scope may be achieved by

implementing the most important activities first, and by gradually supplementing the developed system with automation of other activities – due to its practical importance, this topic is elaborated in a broader extent in the next Subsection 4.7.2.

Next simplification pertains to the extent of SLS *Requirements attributes* and allowed ranges of their values to be supported by the developed automation solution and the underlying Network Slicing orchestration stack. As argued in Subsection 2.3.2, a given Orchestrator is unlikely to support all of the standardized SLS *Requirements attributes*, even after years of development. Moreover, almost all attributes of the `ServiceProfile`/`SliceProfile` are currently defined as optional, making solutions supporting only their chosen subset still fully standards-compliant. As such, the architectural decision may be to focus on providing an automated design for the chosen SLS *Requirements attributes* only and resign from automated processing of the less significant ones, leaving them for potential consideration in a design process conducted manually or in an assisted manner, if passable. Similarly, some ranges of the SLS *Requirements* values can be excluded from automated processing (e.g. stringent values of demanded latency or throughput). In practice, an evolutionary approach towards automated support of the SLS *Requirements attributes* specified in the standardized `ServiceProfile`/`SliceProfile` is expected – from a narrow subset of supported SLS *Requirements attributes* and their narrowed ranges of values, up to the coverage for their target spectrum (which depends on the advancement of the utilized technology) and full ranges of their possible values. A similar simplification is to implement the more extensive (2) *evaluation* and (3) *selection* mechanisms only for the key *Requirements attributes*, but implement a less insightful treatment of the less essential ones.

Another simplifying assumption is not to consider other *Configuration Profiles* of *fully concretized Slicing Entities* (besides the currently configured one) as *candidates* for (2) *evaluation* and (3) *selection*. This allows for omitting the implementation of complex reasoning rules which need to analyze if the reconfiguration of a shared, *fully concretized Slicing Entity* could influence the satisfaction of other sets of *Requirements* already assigned to them. An even stronger assumption which simplifies multiple aspects in the entire design pipeline is to allow for the design of a new NSI/NSSI from *fully concretized Slicing Entities* only. Applying such an assumption allows for omitting the need of automating the (5) *configuration* activity, which is difficult to achieve without imposing other strong assumptions on the processed data, as pointed out in Subsection 4.3.6. Moreover, applying this assumption excludes a potentially large abundance of *Configuration Profiles* of the *not fully concretized Slicing Entities* from being considered as *candidates* in the *evaluation*. It also allows a potentially complicated representation of *Capabilities* in the *Design-time Slicing Entities* not to be implemented. It must be underlined, though, that this assumption entails the need of manual or assisted pre-configuration and pre-provisioning of a pool of *fully concretized Slicing Entities* which are predicted to be needed when an NSI/NSSI allocation request arrives. As a consequence, such an approach limits the possibilities for dynamic optimization of SLS fulfillment and/or consumed resources, and is against the "dynamic slicing" orchestration (described in Subsection 2.3.6).

4.7.2. Roadmap for gradual rollout of design process automation

Introduction of automation in the preparation phase does not need to be performed in an "all or nothing" manner. Taking into account the simplifying assumptions enumerated in the previous Subsection, a **roadmap for gradual rollout of automation** of the design process is proposed by the author, which is based on a set of gradual expanding Automation Scopes (<AS>). The proposed roadmap of ASs establishes tangible milestones for automation development by incrementally extending the range of automated design activities to be implemented. Following such an approach helps to benefit early from the added value of automation and gradually expand the range of covered business use cases.

Figure 33 schematically depicts the proposed Automation Scopes marked with labels and colors: <AS1> with blue, <AS2> with green, <AS3> with purple and <AS4> with orange. The proposed scopes are incremental and backward-compatible, which means that the next <AS> is a superset of the previous one.

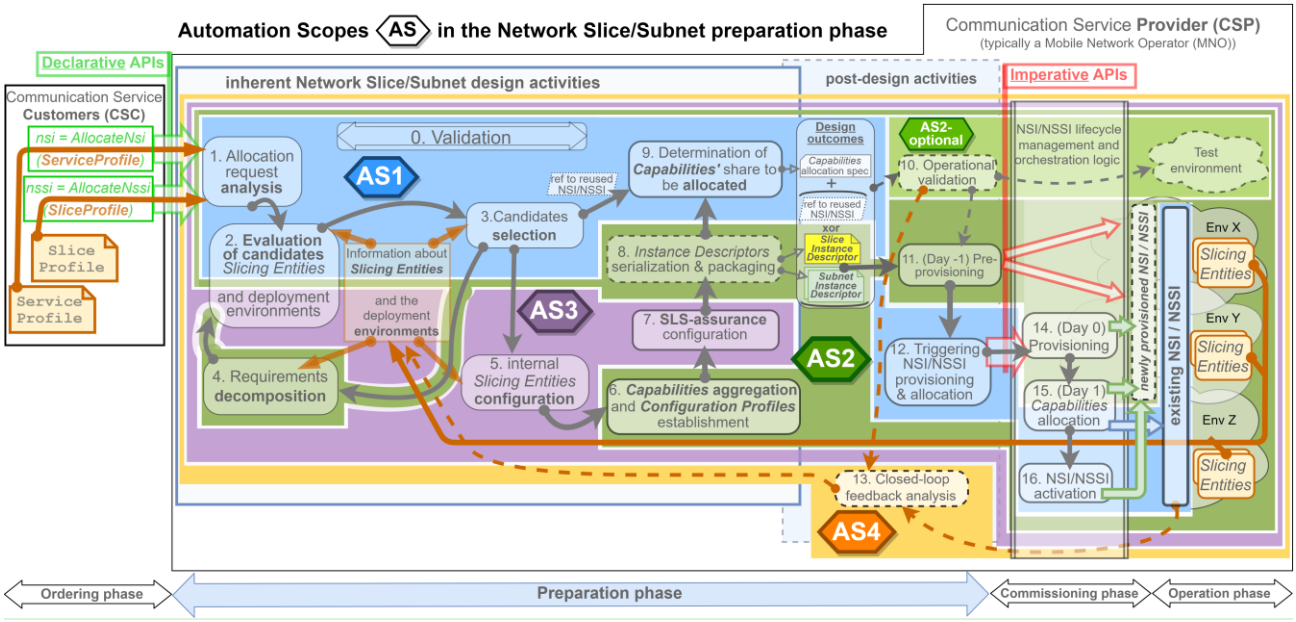


Figure 33. Roadmap of incremental *Automation Scopes* <AS> rollout in the Network Slice/Subnet preparation phase.

In <AS1>, automated allocation of an existing *fully concretized Slicing Entity* representing Slice/Subnet as a whole (pre-designed *Instance Descriptor* of a Slice/Subnet, reserved-but-not-yet-deployed NSI/NSSI or an already running NSI/NSSI) is enabled for the input set of SLS *Requirements*. Realization of <AS1> requires implementation of the following basic aspects: exposition of *Capabilities* and *Configuration Profiles* only by the *fully concretized Slicing Entities* representing Slices/Subnets, as well as implementation of automated (1) *allocation request analysis*, (2) *evaluation* and (3) *selection* activities, which only need to support such types of *candidates*. If there is no suitable *candidate* found, the allocation request is either denied or it is scheduled for manual intervention by the Operator.

It must be underlined that <AS1> entails manual or assisted preparation of a pool of *fully concretized Slicing Entities* representing Slice/Subnet, prior to the Customers' invocations of the `allocateNsi`/`allocateNssi` operations. As such, if the existing pool of such *Slicing Entities* is exceeded, or their offered *Capabilities* do not match the input SLS *Requirements* posed by the Customer, it is not possible to dynamically design and spawn a new NSI/NSSI to meet these *Requirements*. Moreover, utilization of resources in the deployment environments is not dynamically shaped, as all of the design decisions are made earlier and are statically stored in the created *fully concretized Slicing Entities*. Nevertheless, implementation of the minimal set of automated functionalities assumed by <AS1> already allows for supporting a potentially large fraction of `allocateNsi`/`allocateNssi` requests in an unattended manner, enabling coverage of several basic business use cases, e.g. sharing a limited number of pre-provisioned, large scale and heavyweight Slices by multiple CSIs. Moreover, implementing the <AS1> scope puts the necessary foundations for further development of automation mechanisms in the design process.

<AS2> enhances <AS1> by enabling allocation of new NSIs/NSSIs for the incoming SLS *Requirements*, with the assumption that they are designed only from the already existing *fully concretized lowest-level Slicing Entities* (MFs and MLs). Realization of <AS2> requires supplementing the mechanisms implemented in <AS1> with: exposition of *Capabilities* and *Configuration Profiles* by the *fully concretized lowest-level Slicing Entities* and their support by enhanced (2) *evaluation* and (3) *selection* activities, as well as implementation of the automated mechanisms for (4) *Requirements decomposition*, (6) *Capabilities aggregation and Configuration Profiles establishment*, (9) *determination of Capabilities' share to be allocated* and, potentially, (8) *Instance Descriptors serialization and packaging*. Additionally, in order to perform seamless deployment of the newly designed NSI/NSSI, automated logic for the core post-design activities needs to be implemented as well, namely for the (11) *pre-provisioning* as well as for (12) *triggering NSI/NSSI provisioning and allocation* in the target deployment environments. Optionally, <AS2-optional> may be implemented, by automating the (10) *operational validation* activity. Realization of <AS2>, although enabling dynamic creation of new NSIs/NSSIs, still does not offer full dynamicity in shaping utilization of the underlying resources, as the *lowest-level Slicing*

Entities need to be manually prepared in advance by the Operator, prior invocations of the `allocateNsi`/`allocateNssi` operation by the Customers. Nevertheless, implementation of <AS2> allows for gradual introduction and testing of the automated design activities, which are prerequisites for achieving full dynamicity in <AS3>.

<AS3> allows for supporting flexible allocation of the existing or new NSIs/NSSIs. New NSIs/NSSIs are dynamically designed out of a full spectrum of *Slicing Entities*, including the *Design-time* descriptions of the *lowest-level Slicing Entities*, which is needed for dynamic spawning of new MFIs and MLIs. In <AS3>, all the mechanisms implemented in <AS1> and <AS2> are supplemented by: exposition of *Capabilities* by the development environments, enhancement of the (2) *evaluation* and (3) *selection* activities to support development environments as *candidates*, and most notably, implementation of mechanisms allowing automated (5) *configuration* of the *lowest-level Slicing Entities* to be newly instantiated and (7) *SLS-assurance configuration*. By implementing <AS3>, the design process is enabled to flexibly decide whether to reuse the existing *Slicing Entities* or to dynamically spawn new ones. As such, <AS3> allows for dynamic shaping of resources utilization, fully supporting the "dynamic slicing" principles from the perspective of the NSI/NSSI preparation phase.

Finally, <AS4> supplements <AS3> with automation of all the inherent and post-design activities, including the advanced, optional activity of automated (13) *closed-loop feedback analysis*. Implementing such a scope not only enables unattended Orchestrator's operation in the preparation phase, but also automated enhancement of decisions during the design process, allowing self-improvement of the design subsystem.

4.7.3. Desired level of automation

Automation of complex processes can be implemented in various degrees, as classified on a 10-level scale elaborated based on [164], presented in Figure 34. According to this scale, the nature of human Operator's involvement interaction with an automated system can be classified as **manual**, when all activities need to be performed by the human Operator, **assisted**, when the system supports the Operator but leaves some decisions to be made by him/her, and **unattended**, when the execution of the entire process does not require any involvement of the Operator.

Automation level		Description of the level	Trend	Kind of human Operator **involvement**	Is human Operator's intervention **required?**
HIGH	10.	The computer decides everything, acts autonomously, ignoring the human	Increasing autonomy, less reporting about executed functionality	(c) *unattended*	NO
	9.	informs the human only if it, the computer, decides to			
	8.	informs the human only if asked, or			
	7.	executes automatically, then necessarily informs the human, and			
autonomy border	6.	allows the human a restricted time to veto before automatic execution, or			autonomy border
	5.	executes that suggestion if the human approves, or	Increasing involvement of the human operator, more input information required	(b) *assisted*	YES
	4.	suggests one alternative			
	3.	narrows the selection down to a few, or			
	2.	The computer offers a complete set of decision/action alternatives, or			
LOW	1.	No assistance offered: human must make all decisions and select actions		(a) *manual*	

Figure 34. Levels of automation of a computer system, with the specification of human Operator involvement and autonomy border.

Although it may seem that unattended operation of the entire design pipeline in each possible case should be perceived as the ultimate goal to be achieved, an important question should be answered before making such attempts: should all NSI/NSSI allocation requests be indeed handled in an unattended way? An intuitive wish is to possess a fully automated design-time solution which handles all `allocateNsi`/`allocateNssi` requests, no matter what type of Slice/Subnet is requested, no matter what set of SLS *Requirements attributes* and their values are passed in the input `ServiceProfile`/`SliceProfile`, and no matter what the currently experienced conditions in the underlying deployment environments are. However, realization of such a vision not only requires significant development efforts, but also may not be desirable in several circumstances.

In a basic scenario, the Operator may wish to supervise the execution of the automated design pipeline, especially in its initial rollouts. Such a supervised mode of operation executes the design activities

automatically, but interrupts the flow at configured stages to allow the Operator to review and approve the intermediate results (e.g. computed *evaluation* results, *selection* decisions, values of the *decomposed Requirements*, *aggregated Capabilities* values etc.), potentially applying manual corrections. Enabling supervised execution should be possible via configuration of the design process, and should preferably be supplemented by a Graphical User Interface (GUI) that allows convenient overview of the executed process[130]. Once enough confidence in the accurateness and reliability of automated processing in the particular design activities is achieved, the Operator may gradually switch off the supervised mode, finally reaching fully automated, unattended operation.

Another scenario when the operation of automated design pipeline may be deliberately put into one of the assisted levels or even switched off, is when the incoming allocation request is demanding, e.g. in terms of rigorous SLS *Requirements*, requested Slice/Subnet scale, or anticipated costs of its deployment. For example, if the request is to allocate a large-scale Slice/Subnet (e.g. international, continent-wide or world-wide) and/or capable of meeting stringent *Requirement* value on the end-to-end latency (e.g. 5ms), then even though such a request could be handled in an unattended way, the design pipeline is deliberately configured to execute in a supervised mode. This is because handling such a demanding request may entail significant expenditures (e.g. dedication of the most powerful resources or enablement of their advanced features, moving workloads to the edge, etc.), and, as such, additional review of the design outcomes by the Operator may be required before its acceptance. Besides the aforementioned scenarios, a variety of business (e.g. profitability of accepting a particular request or Customer's business status), political (e.g. military or governmental Slices/Subnets), lawful (e.g. requirement to deploy specialized deep packet inspection functions, or to enable lawful interception features) or organizational factors (e.g. planned maintenance, expansion of covered areas, etc.) may entail the need to decrease the level of automation in the Network Slice/Subnet preparation phase.

In practice, a **hybrid approach** is expected to prevail, in which the frequently occurring, well-supported equivalence classes of allocation requests are handled in an unattended way and the ones containing more demanding values of SLS *Requirements* require either manual/assisted handling, or the automatically prepared design outcomes require explicit approval by the Operator before their operational utilization. It is worth noting that awaiting such approval does not need to break the autonomous operation of the automated pipeline – the system can be configured to notify the Operator about the arrival of a demanding request (e.g. by mail, SMS, etc.), containing the details of automatically prepared design outcomes and allowing for a limited amount of time (e.g. 5 minutes) to object its utilization. If no objection is sent within the given deadline, the automated system proceeds with operational utilization of the design outcomes, guaranteeing unattended, semi-autonomous operation of the preparation phase pipeline.

4.7.4. Design process duration and dynamicity

Due to its extent and complexity, the process of designing a Network Slice/Subnet is not an instant operation. If all the design activities assumed by the discussed framework are performed manually, the process may take minutes, hours or even days, and be conducted in multiple, interrupted timeslots until the finalized outcome is produced. Execution of an assisted design process, depending on the level of automation, will usually take minutes and, due to the need of providing manual input at some of its stages, may also be performed in interrupted timeslots. Finally, the fully automated, unattended design process is expected to take no more than a few seconds of continuous, uninterrupted operation.

Over the course of the design process, regardless of its duration, changing state of the considered *Slicing Entities* and variable conditions in the target deployment environments may be observed. For example, a particular *Slicing Entity* might have been present in the environment at the beginning of the design process, but is no longer available at its end; or, the current load of the deployment environment has increased significantly from the start of the process, etc. These and many other examples of **deployment environments' dynamics** undoubtedly influence the way in which the design process shall be implemented. Depending on

[130] The supervised mode of operation may be also utilized, for example, for training a neural network that would support automated decisions-making in the design process. Utilization of AI techniques in the Network Slicing design area is discussed in Section 7.4.

the architectural decisions and functionalities offered bythe Orchestrator, a variety of approaches may be utilized to handle these issues.

If the integration between the design subsystem and the Orchestrator is tight, the design process may, for example, utilize a kind of a "temporal lock" to create short-lived reservations on the *Slicing Entities* considered in the designed NSI/NSSI and the resources they need. However, such an approach effectively introduces the need of handling state of the design activities on the side of the orchestration logic. Thus, such an approach may complicate the Orchestrator's operation by limiting the flexibility of the LCM procedures it performs and introduce problems of handling unfinished designs when multiple parallel design processes are being conducted, e.g. as a result of spontaneously invoked `allocateNsi`/`allocateNssi` operations in a Network Slicing solution supposed to support "dynamic slicing" orchestration capabilities.

For the aforementioned reasons, in the described framework it is assumed that the design process does not induct any state in the Orchestrator. As such, the design activities are only allowed to trigger read-only operations in the Orchestrator's subsytems, e.g. querying their operational inventories, requesting PM/FM data, etc. Such an approach simplifies the interactions between the design process and the Orchestrator, and it also removes the need of handling the aforementioned problems related to maintaining the design state inside of the LCM logic. In this context, it must be noted that the design activities are performed based on the environmental conditions probed at a particular moment in time, assuming this data is valid until the end of the process. The consequence of this assumption is that the Orchestrator should perform a *pre-instantiation validation* (see Subsection 4.6.3) of the design outcomes at the moment of triggering the NSI/NSSI provisioning and allocation procedures, in order to verify if the design decisions conveyed therein are possible to be realized in the currently experienced conditions, or are "acceptably different" to proceed. Lastly, the automated individual design activities and the entire design process may be leveraged during the operation phase for performing continuous, dynamic adaptation of the NSIs'/NSSIs' structure as a part of the resources capacity (re-)planning.

4.8. Summary

In this Chapter, the course of the end-to-end allocation process has been explained, by decomposing it into fine-grained, individual activities spanning the preparation, commissioning and operation phases of the NSI/NSSI lifecycle. The author focused on analyzing the logic of the design and post-design activities, describing how each of them can be automated by leveraging the relevant pieces of the captured domain and operational knowledge conveyed in the modeling constructs presented in Chapter 3. Based on the conducted analysis, the author established a coherent and generic conceptual framework that paves the way for constructing systems capable of performing automated, fully standards-aligned Network Slice/Subnet design process supporting the *declarative* NSI/NSSI *provisioning style*.

The established conceptual framework promotes structurization of the design activities into a set of **fine-grained**, **coherent** and **generic, single-responsibility procedures** with **clear input/output** and **complementary roles**. Following these principles results in the proposition of a range of simple methods that can be flexibly composed into various arrangements, facilitating the formulation of automated design workflows adjusted to the features and operational scope of the Orchestrator being augmented. In particular, automation of the design process can be rolled out gradually by implementing the backward-compatible Automation Scopes proposed in this Chapter. Thanks to its generic nature, the presented framework is open for extensions and modifications, allowing for its alignment both to the characteristics of the Orchestrator being augmented (its supported operational APIs, availability of operational data, etc.) as well as to the changes in the evolving Network Slicing standardization.

In the following Chapter, the author proves that by utilizing the constructs of the elaborated framework, automation of the design process in a selected variant can be achieved in a sample Orchestrator which before augmentation only supported the *imperative provisioning style*.

Chapter 5

Proof of Concept implementation

In the previous Chapters, a broad range of concepts comprising the Network Slice/Subnet design automation framework have been introduced and comprehensively explained. In order to prove their feasibility and, most importantly, to verify the thesis statements, the key elements of the framework need to be embodied in a system that can be empirically investigated. In this Chapter, PoC implementation details are presented, showcasing how the embodiment of the elaborated concepts enabled automation of the design process in a sample Orchestrator (here and thereafter referred to as "Subject Orchestrator"), resulting in the support of fully automated design and allocation according to the *declarative style*.

Firstly, in Section 5.1, the PoC assumptions and expectations are presented. In Section 5.2, the choice of a Subject Orchestrator is discussed. Then, in Section 0, selected technical details of the Subject Orchestrator are presented, including its original architecture and the operational scenarios it originally supported. Afterwards, in Section 5.4, the operational shortcomings of the Subject Orchestrator that are targeted to be improved in the PoC are discussed. Next, in Section 5.5, the implementation details of individual models and activities are presented. In turn, in Section 5.6, it is described how these individual elements have been combined into a coherent, fully automated end-to-end allocation workflow that is aligned with the *declarative provisioning style*. Finally, a summary of the PoC implementation aspects is provided in Section 5.7, where the achieved functional goals are highlighted and several conclusions related the technical realization of the proposed automation framework are stated.

5.1. PoC assumptions and expectations

In this Section, basic assumptions and expectations of the PoC are described, determining the context of the discussions in the remaining content of this Chapter and in Chapter 6. Firstly, the research goals and methodology adopted in the PoC are clarified. Afterwards, scope of the PoC is specified. Finally, several important PoC caveats are stated.

5.1.1. PoC goals and research methodology

The main goal of the PoC is to verify the thesis formulated in Section 1.4, which states that appropriate modeling of the domain&operational knowledge and of the Network Slice/Subnet building blocks, as well as appropriate functional structuralization of the design process are expected to enable its automation. Thus, the **research methodology** adopted in the PoC assumes **functional augmentation** of a Subject Orchestrator, which originally required manual design process performed according to the *imperative provisioning style*, with the feature of performing fully automated design and allocation according to the *declarative style* principles.

Experimental verification of the thesis statements must show not only that the implementation of the automation framework concepts is **feasible**, but mainly that their embodiment in a Subject Orchestrator **indeed enables automated design process execution that excludes the need for human Operator's participation** in handling the allocation requests. Hence, the PoC solution's operation shall be showcased in conditions closely resembling real Network Slicing orchestration in a given domain. Implementation of the PoC is also expected to practically showcase the fulfillment of the remaining requirements specified in Section 2.4 – in particular, the coherency and genericity of the proposed framework, but also that a system constructed based on its concepts is multivariant and extendable.

Furthermore, in order to gain a more comprehensive insight into the operational characteristics of the developed solution, its experimental evaluation should also be performed from a **non-functional perspective**. In this context, the goal of conducting **performance evaluation** is to verify that the developed solution functions correctly in increasingly demanding operational conditions and to assess how well does it perform. In particular, the obtained results should indicate if the elaborated concepts may indeed support on-demand allocations and support operation of the Subject Orchestrator according to the "dynamic slicing" principles. Furthermore, by collecting a spectrum of performance measures, the solution's operational limits can be identified, revealing its bottlenecks and areas requiring improvements. Based on the overall evaluation results, judgement of operational maturity and industrial applicability of the elaborated concepts may be attempted as well.

It shall be noted that the content of the current Chapter is devoted to the functional augmentation research aspect of the PoC - explanation of the implementation details presented hereafter allows for verifying the feasibility and coherency of the proposed concepts. In turn, the upcoming Chapter 6 is dedicated to the description of the empirical investigations of the PoC solution, aiming at verifying the achieved functionalities and assessing its performance.

5.1.2. PoC implementation requirements and scope

All the concepts comprising the automation framework proposed in this thesis have been elaborated with alignment to the 3GPP Network Slicing provisioning standards as their core value. Thus, as a primary requirement, the PoC implementation should be fully aligned with the 3GPP standardization, which dictates that the NSI/NSSI allocation procedure is expected to be **end-to-end** in its nature. This means that it begins at the moment of obtaining the Customer's SLS *Requirements* and shall only finish after an NSI/NSSI that is fully configured to support them is up and running.

At the implementation level, such an end-to-end allocation procedure can be realized as a **workflow** in which the selected activities assumed in the conceptual framework are grouped into five phases schematically depicted in Figure 35: (A) "Allocation request analysis", (B) "Determination of a *Slicing Entity* representing Slice/Subnet", potential (C) "New NSI/NSSI design" followed by (D) "New NSI/NSSI provisioning", and finally (E) "NSI/NSSI re-configuration for `ServiceProfile`/`SliceProfile` allocation" according to the design decisions.

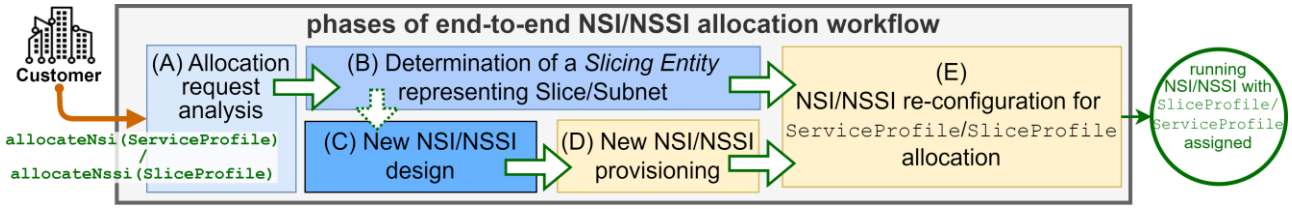


Figure 35. Overview of coarse-grained implementation-level phases of an end-to-end NSI/NSSI allocation workflow.

While all the inherent design activities take place in phases A, B and C (marked with blue background), the provisioning and allocation procedures performed in phases D and E (marked with yellow) are merely about applying the design decisions produced in the previous phases. Thus, the main focus of the PoC is on automating the processes in phases A, B and C. On the way towards verifying the thesis statements, the details of processing in the enumerated phases should be juxtaposed between the original features of a Subject Orchestrator and its features after PoC implementation. This will allow for assessing the achieved functional augmentation, clearly depicting the added value brought by the application of the automation framework's concepts.

To serve as an experimental platform for thesis verification, the PoC implementation scope concerns selected elements of the elaborated framework that are essential for achieving design activities automation in phases A, B and C, and their inclusion into an end-to-end allocation workflow. The necessary development works include:

1. implementation of selected ***Slicing Entities*** **models** with **uniform exposition of *Capabilities***, assuring their correspondence to the SLS *Requirements attributes* supported by the Subject Orchestrator in the allocation requests[131];
2. implementation of **models conveying the domain, operational and business knowledge** needed for conducting the essential design activities in an automated manner. The implemented models need to convey, at least:
   a) specification of *Slicing Entities*' *evaluation* methods, dictating how the particular *Capability-Requirement* pairs should be compared and matched;
   b) specification of *Requirements Decomposition Rules* (RDR), dictating the correspondence between the higher-level and lower-level *Requirements attributes*;
   c) specification of *Capabilities aggregation* methods, dictating how the lower-level *Capabilities attributes* are merged into the higher-level *Capabilities attributes*;
   d) specification of *Slicing Entities*' *selection* policies reflecting the Operator's and Provider's operational and business objectives;
3. implementation of **logic for the essential design activities** in phases A, B and C, capable of performing automated (1) *allocation request analysis*, (2) *evaluation*, (3) *selection*, (4) *Requirements decomposition*, (6) *Capabilities aggregation* according to the domain&operational knowledge conveyed in the aforementioned models, as well as (9) *determination of Capabilities' share to be allocated*;
4. implementation of logic for the necessary post-design activities to execute automated (11) *pre-provisioning* and (12) *triggering provisioning & allocation*;
5. implementation of a **coherent workflow** combining all the mentioned models and activities into an end-to-end automated pipeline that excludes the need of human participation in handling the allocation requests.

In fact, the list above contains the elements which a Slicing orchestrator needs to support <AS2> described in Subsection 4.7.2. Indeed, <AS2> encompasses the core of the proposed framework, essential for enabling automated design process that supports hands-free allocation of both existing NSIs/NSSIs and the ones to be newly designed and provisioned. As such, implementation of <AS2> enables automated allocation in the basic operational scenarios, making it sufficient for the experimental verification of the thesis statements. Implementation of <AS1> only would be too limited as it concerns the allocation of existing NSSIs only, omitting the models and logic that enable the design and provisioning of new NSIs/NSSIs.

On the other hand, implementation of the more advanced <AS3> or <AS4> scopes, although enhancing the design process with desirable features[132], revolves around technical refinements of the core <AS2> elements and inclusion of mechanisms for supporting advanced operational cases. Therefore, while the implementation of <AS2> is essential for **enabling** automated design process execution, which is the key statement to be verified in this thesis, realization of <AS3> or <AS4> is about its enhancements. Therefore, implementation of the advanced elements of <AS3>/<AS4> is no longer contributing towards thesis verification and, as such, is considered out of scope of the PoC. However, as it is discussed in the upcoming Subsection 5.7.3, implementation of <AS3> and <AS4> is considered as a future work, based on which further research related to the design process can be conducted.

A more detailed list of limitations of the implemented PoC solution with respect to the holistic scope of the generic conceptual framework presented in Chapters 3 and 4 is presented in the upcoming Subsection 5.7.3, along the PoC development summary.

5.1.3. PoC caveats

Before diving into the PoC details, several important caveats need to be stated in order for the reader to distinguish the essence of the PoC from the aspects which, although described in its course, should not be perceived as its gist.

[131] The set of supported SLS *Requirements attributes* depends on the operational scope of the Subject Orchestrator in which the PoC is implemented and the features of *Slicing Entities* it manages. This important aspect is further discussed in the upcoming Subsections 5.4.3 and 5.6.2.

[132] e.g. consideration of *Design-time Slicing Entities* for dynamic spawning of new *lowest-level Slicing Entities* instances or the (13) *closed-loop feedback analysis* activity – see Subsection 4.7.2 for the description of <AS3> and <AS4>.

The concepts presented in this dissertation and, in particular, their PoC implementation are about the **organization** of models and methods that allow for **embedding and leveraging the domain and operational knowledge** for enabling automation of the **design process**. As pointed in the specification of the scope of this thesis (see Section 1.6), only the representation of this knowledge, the way of its expressing and its processing methods are the subjects of this dissertation and are experimentally verified - **not the conveyed knowledge itself**. More specifically, in the conducted experiments, all models conveying this knowledge are configured with particular sets of values, determined to the best of the author's expertise to reflect the conditions of Network Slicing management in the particular operational circumstances. However, even in the same conditions as assumed in the experiments, such values could be configured differently by various Operators supervising the Subject Orchestrator's operation. Thus, the utilized pieces of expert knowledge **serve merely as examples** – **they do not constitute a subject of verification** and should not be perceived as an operational reference.

It also shall be noted that the PoC solution has been developed purely for experimental purposes and should by no means be considered a production-grade system. As such, both the implemented functionalities and their experimental testing are focused on the aspects that are essential from the functional research perspective, without delving into a range of highly-specific corner cases that would need to be considered in the development of a production level system[133].

5.2. Choice of a Subject Orchestrator

For showcasing the realization of the assumed PoC scope, a Subject Orchestrator is needed. Implementing a Network Slicing Orchestrator from scratch is a challenging and long-lasting effort that would require significant resources to be accomplished, which is beyond the capabilities of the author alone. Moreover, such efforts would not directly correspond to the subject of this thesis and, as such, are out of consideration. Therefore, the author decided to perform a survey of existing orchestration solutions[134] that could serve as an experimentation platform in the PoC.

5.2.1. Preliminary remarks and survey assumptions

Before presenting the results of the survey, it must be noted that at the moment of writing this thesis the inception of the Network Slicing technology, including the development of the corresponding Orchestrators, is still at its early stage. The currently available solutions are far from offering all the use cases assumed in the 5G visions and they usually do not support the full spectrum of models and APIs dictated by the dynamically evolving standards. Instead, such systems usually cover their limited subsets, with plans of their implementation in the long-term development roadmaps. At the technical level, this fact manifests itself in various operational limitations as well as simplifying architectural and implementation assumptions that have been noticed during the conducted survey.

For the aforementioned reasons, the choice of a Subject Orchestrator needs to yield several assumptions. The survey has been focused on selecting an Orchestrator which is in a development state that allows implementing the assumed PoC scope, limiting the need of delving into side issues, tackling of which could take the author far afield from the primary goals of the experimentation. Thus, the modeling approach utilized in the considered projects has been taken into account, as well as the already offered APIs and supported kinds of *Slicing Entities*. Importantly, since the concepts of the automation framework elaborated in this thesis have been developed with the 3GPP standards-alignment treated as their core value, an orchestrator built in a 3GPP standards-compliant manner is a preferred choice.

5.2.2. Considered open-source Orchestrators

Initially, the author considered utilizing one of the open-source Orchestrator for the implementation of the PoC. There are several publicly available projects - each developed according to different conceptual, architectural and operational assumptions. The majority of the research projects developed in the public-private consortiums have been built at the early stages of the Network Slicing concepts formulation, e.g.

[133] Potential for industrial applicability of the concepts elaborated in this thesis is discussed in the upcoming Section 7.2.
[134] The survey has been conducted in March-April 2022, before kicking-off the implementation works.

5GPagoda [127] in years 2016-2019, 5GNorma [165] in years 2015-2017 or NECOS [166] in years 2017-2021. At these timeframes, the standardization of the provisioning models and APIs was not yet attempted or was in its initial stages. Due to this fact, the organization of the orchestration architectural blocks, provisioning approaches, APIs and models utilized in these projects do not follow the currently acclaimed, standardized principles. Additionally, the source codes of the public-private projects are not always freely available as the works on their development is restricted to the members of the consortium and the majority of these projects is not actively developed anymore.

Among the actively developed open-source Orchestrators the choice is limited [142][123]. The scope of these projects differs and their degree of compliancy with the provisioning standards is varying or, sometimes, not targeted. As an example, in the Open Source MANO (OSM) [123], features of which have been already partially discussed in Subsection 2.5.2, a customized set of non-3GPP-compliant models and APIs are utilized - there is no notion of `ServiceProfiles`, `SliceProfiles` or `ManagedFunctions`. Moreover, shareability of the Slice/Subnet building blocks and management of the SLS *Requirements* is also not addressed by OSM. Thus, these abstractions would need to be implemented in OSM from scratch. Moreover, the notion of *Slicing Entities*' and their *Capabilities*, against which *Requirements* could be posed, would also need to be implemented from scratch. Additionally, the provisioning APIs, instead of implementing the 3GPP-standardized signatures (`allocateNsi`/`allocateNssi`), are utilizing a variation of the ETSI NFV-SOL005-inspired, imperative operations for "Slice" provisioning. Reworking the OSM solution to adhere to the 3GPP standards would require a large dose of architectural and development works, not directly bound to the assumed PoC scope.

ONAP [142] was also taken into account to be utilized in the PoC, since its standards-alignment has been considered with increasing attention by the project community in the recent years. However, despite the versatility of the envisioned architectural variants presented in [143], until now, only two out of five (variants 1 and 4) have been targeted in the actual development. In these variants, the ONAP-based NSMF and NSSMF blocks instead of exposing the `allocateNsi`/`allocateNssi` provisioning operations as the system's NBI, hide their execution as internal calls. Additionally, these operations do not strictly follow the signatures assumed by the standardized operations[135]. Besides the API-level issues, utilization of ONAP for PoC would entail significant efforts on the modeling level, because there is currently no notion of *Slicing Entities* and their *Capabilities* with semantics resembling the concepts presented in this thesis[136]. As a last point, ONAP is a large and heavyweight platform that aims at supporting not only the Network Slicing functionalities, but also a large spectrum of other use cases implemented by various parties participating in the project consortium. This resulted in a high technological complexity, which for efficient development requires close collaboration within the project consortium, as well as a direct involvement in a wider task force for obtaining appropriate support and accesses (e.g. to source code, development environments, CI/CD pipelines, etc.).

As detailed above, the conducted survey revealed several conceptual and technical issues in the current state of the open-source solutions, which could pose serious obstacles for the PoC development in the assumed scope, by the author alone. While the PoC realization in the indicated projects is possible, at the implementation level it would require long-lasting preparatory works of a team of developers before the presented concepts could be experimentally verified.

5.2.3. Possibility of utilizing a proprietary Orchestrator prototype

Thanks to the author's direct involvement in the development of a proprietary Network Slicing orchestration solution and the courtesy of the company which allowed its utilization in the PoC experimentations, the choice of a Subject Orchestrator did not have to be limited to the open-source solutions only. After considering the possible choices, the author decided to utilize the non-commercial prototype of the **Samsung's RAN Subnet Orchestrator**, to which the author contributed as an architect while working at the Samsung R&D Institute Poland. The decision to utilize the Samsung's prototype is justified by several factors discussed in the paragraphs below.

[135] Besides `ServiceProfile`/`SliceProfile`, an explicit pointer to the NST/NSST defining the NSI/NSSI structure must be passed in the internal call.
[136] *Capabilities* are not exposed by all classes and lifecycle stages of entities utilized in ONAP that can be classified as *Slicing Entities*.

Firstly, the author is familiar with the technological stack and principles applied for the Samsung's prototype development. Additionally, the author receives the necessary support in terms of hardware infrastructure for establishing the experimental testbeds, their maintenance by dedicated Development&Operations engineers (DevOps) and the development support from the Samsung's team members when necessary. Secondly, thanks to the author's contributions, among others, the architecture of the Samsung's prototype has been already largely aligned with the concepts proposed in this thesis. Thirdly, the operational scenario supported by the Samsung's prototype (described in the upcoming Subsection 5.3.2) allows for the realization of the targeted <AS2> scope, matching well the assumed scope of the PoC.

There are also other important determinants supporting such a decision. The 3GPP standards-compliancy of the Samsung's prototype, both at the modeling and APIs level is targeted to be realized in full extent, and the development of the PoC is actually contributing towards achieving this goal[137]. Furthermore, the operational stability of the Samsung's prototype is high, which allows for limiting the time spent on debugging auxiliary issues that could handicap the PoC development. As the last factor, the operational maturity of this project is higher in comparison to the open-source solutions, as its functioning with real equipment (prototypes of Samsung's gNB MFs) is verified as a part of the operational tests.

Despite many advantages that speak in favor of utilizing the Samsung's prototype, there are also several shortcomings that have been taken into account by the author. Due to the research and development of not only the RAN Subnet Orchestrator but also of the *Slicing Entities* it manages (gNB MFs: RU, DU, CU-CP, CU-UP and MLs between them), which was ongoing during the author's works, the prototype in its original scope offers support for a relatively limited set of SLS *Requirements attributes* – this issue is discussed in the upcoming Subsection 5.4.3. The prototype in its original scope also misses integration between some of its internal components, which prohibits immediate introduction of the more advanced Automation Scopes: <AS3> and <AS4> - this aspect is described more broadly in Subsection 5.3.2 and in the context of future works in Subsection 5.7.3.

Nevertheless, by taking all of the discussed factors into consideration, the author decided to implement the PoC within the Samsung's RAN Subnet Orchestrator prototype, naming the developed PoC system **"Automated RAN Subnet Design" (ASD)**. Such a choice allows the author to focus the implementation efforts on the merits of the PoC instead of the long-lasting preparatory works of side relevance that would need to be performed in the pointed open-source projects, making the successful realization of the PoC assumptions much more probable. As a final remark, it is important to underline that the PoC implementation inside of a Subnet-level Orchestrator does not limit the genericity of the elaborated concepts and their applicability in a Slice-level Orchestrator - thanks to the 1-to-1 relation between the Slice and Subnet entities (discussed in Subsection 2.3.1), the technical design of a Slice corresponds to the design of a distinguished end-to-end "top" Subnet. Thus, implementation of the PoC in the context of a Slice-level Orchestrator would utilize analogous models and processing mechanisms.

5.3. Overview of the Subject Orchestrator, before PoC implementation

In this Section, selected technical details of the Subject Orchestrator in its original form are described, including its architecture (Subsection 5.3.1), the operational scope it supports (Subsection 5.3.2) and the details of the MF *Slicing Entities* it is capable of managing (Subsection 5.3.3). Presentation of these aspects clarifies the technical and operational context in which the PoC is realized. In the upcoming Sections, it serves as a background for highlighting the automation framework's elements implemented within the Subject Orchestrator's structure.

5.3.1. Subject Orchestrator's original architecture

Figure 36 depicts the original architecture of the Subject Orchestrator, before PoC implementation. The presented architectural view consists of 4 major parts: (1) the Network Slicing management blocks and (2) the NFV-MANO blocks which constitute the Subject Orchestrator core components (surrounded by a green

[137] As described in the upcoming Section 5.5, the implementation of the ASD PoC resulted in full compliance of the allocateNssi operation with the 3GPP specification, which was not the case in the original operational scope of the Subject Orchestrator as described in Section 5.4.

rectangle in Figure 36), as well as the (3) systems with which the Subject Orchestrator interacts in the southbound and the (4) actors interacting with the Subject Orchestrator from the northbound. In the following paragraphs, the main responsibilities of the enumerated parts are briefly described. Providing this overview is important for a proper understanding of the technical issues described in the upcoming Sections and the experimental evaluation of the PoC discussed in Chapter 6.

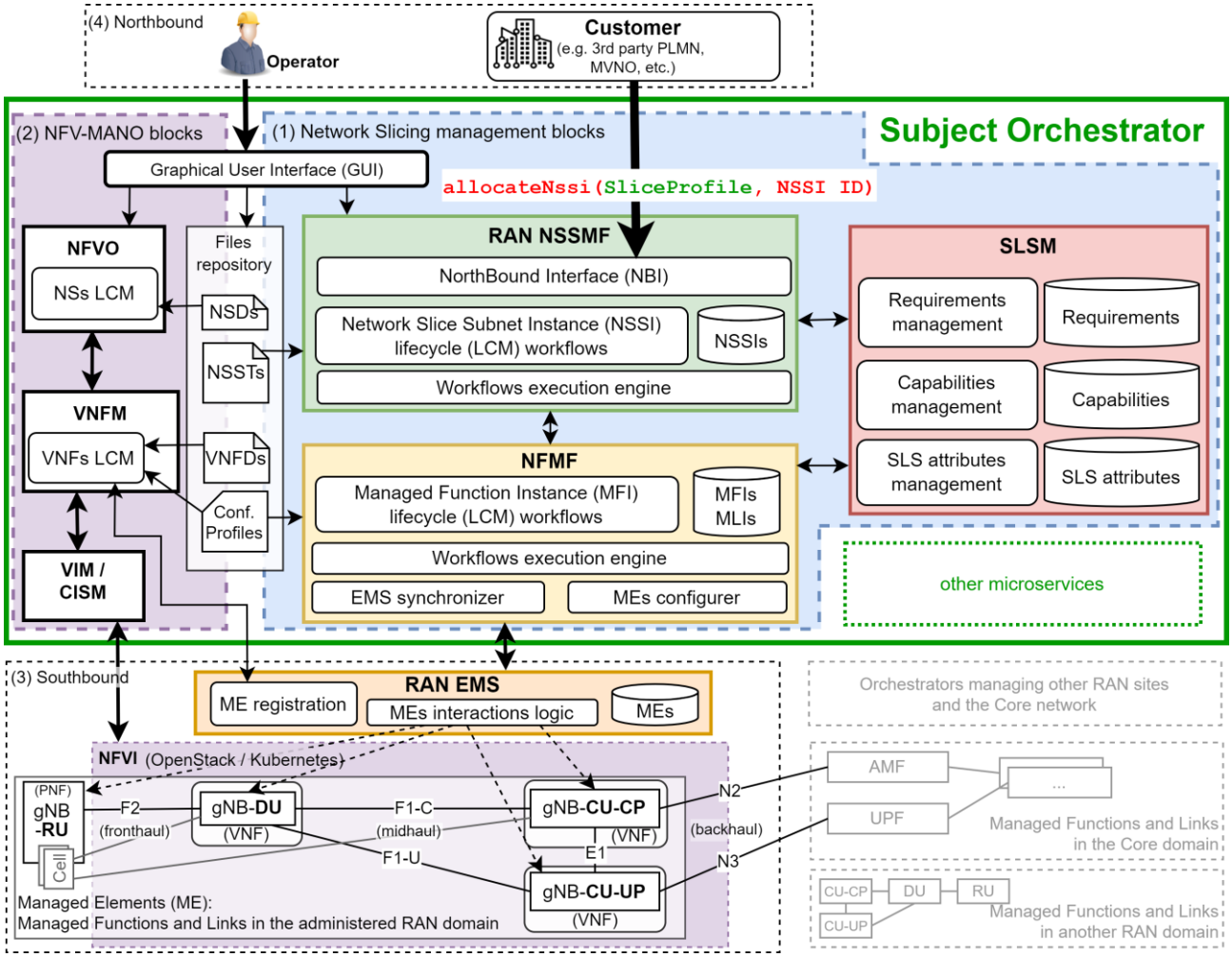


Figure 36. Architecture of the Subject Orchestrator before PoC implementation.

The (1) Network Slicing architectural blocks (contained within the blue shape) consist of the following components: the RAN NSSMF (marked in green), NFMF (marked in yellow) and the SLS Manager (SLSM, marked in pink). The RAN NSSMF microservice is an implementation of a subset of functionalities defined in the 3GPP technical reports [9], [10] and standards [31], [71], [73] related to Subnets management in the RAN domain. The main functionality of RAN NSSMF is the LCM of NSSI objects. In the original operational scope, a given NSSI is created based on an *Instance Descriptor* crafted by the Operator in an assisted design process[138], by selecting and concretizing one of the NSSTs available in the Files repository. RAN NSSMF performs the LCM operations asynchronously, utilizing an internal workflow execution engine.

SLSM is a crucial microservice responsible for the management of *Slicing Entities*' *Capabilities* and the sets of SLS *Requirements* assigned to them. SLSM is a generic component, capable of managing *Capabilities* and SLS *Requirements* regardless of their positioning in the hierarchy, i.e. high-level `ServiceProfiles` against Slices, lower-level `SliceProfiles` against NSSIs as well as lowest-level SLS *Requirements* against the MFIs and MLIs[139]. Among a range of corresponding functionalities, SLSM tracks the *Slicing Entities*' initial and current

[138] Sections 5.4.1 and 5.4.2 describe how the design process and NSSI allocation procedures looked like originally in the Subject Orchestrator, before ASD PoC implementation. Subsection 5.5.2.5 describes the modeling details of the NSST and its enhancements applied as a part of the PoC.

[139] Due to the operational limitations of the Subject Orchestrator, and the TN aspects not being in the scope of this thesis (see Section 1.6), the MLIs' *Capabilities* are empty in the current state of the Subject Orchestrator, but are considered to be supported as a future work. For details, see the `ManagedLink` model implementation described in Subsection 5.5.2.2 and future works description in Subsection 5.7.3.

Capabilities, calculates their offered values, aggregates the sets of SLS *Requirements* and evaluates their fulfillment by the given set of *Capabilities*.

The NFMF microservice is responsible for the lifecycle management of MFIs and stores the information about the MLIs established between them. NFMF provides an abstraction layer between the RAN NSSMF and the underlying, lower-level management systems: the EMS (marked in orange) and the VNFM, which are described in the next paragraphs. NFMF's role is to abstract away the low-level aspects of MEs management in EMS and of the underlying VNFs in VNFM, with the aim of exposing their combined, high-level operational view towards the RAN NSSMF as MFI *Slicing Entities*. To serve such a role, the NFMF's responsibilities include maintenance of MFI objects, equipping them with the abstraction of *Capabilities*[140] (managed inside of SLSM) as well as tracking and managing their Slicing-related configuration[141]. The NFMF's southbound role is to translate the execution of the incoming high-level LCM operations triggered from the RAN NSSMF into the low-level ME-specific commands executed via EMS.

The Subject Orchestrator also includes the architectural blocks comprising the (2) NFV-MANO stack (marked in Figure 36 with a purple background): NFVO responsible for NSs management, VNFM responsible for VNFs management, as well as the VIM and CISM that allow the deployment of VM-based and container-based VNFs (CNFs[23]) into the underlying NFVI environments - OpenStack or Kubernetes, respectively[142].

In the (3) southbound, the Subject Orchestrator interfaces with EMS, through which the interactions with the MEs in the RAN infrastructure are performed. EMS's role is to register, de-register and interact with the control-plane of the deployed MEs, serving as an operational gateway towards the low-level MEs functionalities. EMS manipulates the MEs' attributes' values and executes the operations defined in their corresponding YANG models via the RESTCONF or NETCONF protocols. Among others, EMS forwards the configuration and LCM operations received from the NFMF towards the MEs and monitors their current state in terms of configuration, faults and performance (CM/PM/FM).

At the lowest layer of the architectural view, there are the gNB MEs[143], which in the deployment environments are running either as physical functions (PNFs) or as VNFs (VM-based or container-based). The gNB's Radio Unit (gNB-RU) is a PNF-based ME, whereas the gNB's Distributed Unit (gNB-DU) as well as gNB's Centralized Unit Control Plane (gNB-CU-CP) and User Plane (gNB-CU-UP) are VNF-based. gNB-DU and gNB-CU-CP are managing the radio resources in the Cells that are covered by a given RU. The enumerated gNB MEs are interconnected by fronthaul (F2-C, F2-U) and midhaul (F1-C, F1-U, E1) MLIs. gNBs are connected via backhaul N2, N3 MLIs to the MFIs in the Core domain: AMF and UPF, respectively. The gNB MEs may also be connected via Xn-C MLIs with the gNBs in the other RAN administrative domains (presumably managed by other instances of the Subject Orchestrator or by other Orchestrators, see lower-right part of Figure 36).

In the (4) northbound, the Subject Orchestrator exposes an NBI to be utilized by the Customers. The NBI's standards-compliancy deficiencies (described in the upcoming Subsection 5.4.2) are addressed by the ASD PoC implementation, which is discussed in Section 5.6. The Subject Orchestrator also features a GUI, utilized for convenient management of the entire orchestration subsystems stack by the Operator.

5.3.2. Subject Orchestrator's operational scope and limitations

The main operational scenario targeted by the Subject Orchestrator is to offer RAN NSSIs management for 3rd party PLMN Customers (e.g. MVNOs). The NSSIs management supported by the Subject Orchestrator is performed on top of the **already existing, operationally deployed gNB installations only**. Such operational scope is appropriate for Providers in the initial stages of the Network Slicing technology adoption, in which it is desired that the deployment of the RAN elements is decided and manually controlled by the Operator before their utilization for Slicing. Technically, this means that the *lowest-level Slicing Entities* comprising the gNB installations, namely the MFIs (RU, DU, CU-CP, CU-UP instances schematically shown in the bottom of Figure 36) and their interconnecting MLIs (instances of the F2-C, F2-U, F1-C, F1-U, E1, N2, N3 Links) need to

[140] MFI *Slicing Entity* = ME managed in EMS + VNF managed in VNFM + *Capabilities* managed in SLSM.
[141] These are the values for attributes recognized as "Slice-influenced" and "Slice-specific" in Subsection 3.3.2.
[142] Support for both VIM and CISM stems from the fact that the Samsung's gNB VNFs are offered both as VMs and containers.
[143] Subject Orchestrator only supports gNB deployments in the 5G SA mode. Inclusion of pre-5G RAN MFs (i.e. eNB, en-gNBs, etc.) is not supported.

be *fully-concretized Runtime Slicing Entities* - they **must be instantiated and supplied with bootstrap configuration prior their utilization for Slicing**. Only then, allocation of NSSIs on top of them becomes possible, which is originally performed manually, but becomes fully automated after PoC implementation.

It shall be noted that in the described operational scope, new elements of the gNB installations are not allowed to be spawned autonomously by the Subject Orchestrator. In other words, if there is a need to include new instances of the *lowest-level Slicing Entities* in the NSSIs that are anticipated to be allocated (e.g. spawn new gNB MFIs and MLIs in the not-yet-covered geographical areas), they need to be instantiated and provided with bootstrap configuration by the Operator. This is performed by the Operator in an assisted manner by utilizing the tooling of the NFV-MANO and EMS components [144], before the Customer's `allocateNssi` request considering such elements can be handled.

Although the described assumptions limit the dynamicity of Network Slicing management performed by the Subject Orchestrator (because new MFI/MLI instances cannot be spawned without the supervision of the Operator), they match well the scope of <AS2> which assumes that only the *fully concretized lowest-level Slicing Entitie*s are utilized in the design process. In particular, <AS2> allows for both new designing and allocating new NSSIs, as well as the reuse of the already existing NSSIs. Thus, the Subject Orchestrator's operational scope is in-line with the expected scope of the PoC, making it an **appropriate platform for PoC implementation and its experimental evaluation**.

As a final remark regarding the operational scope, the assumptions described above resulted in an architectural simplification applied during the Subject Orchestrator's development at its prototyping stage, which is important in the context of the anticipated, potential implementation of the more advanced Automation Scopes. Although the NFV-MANO architectural blocks are a part of the Subject Orchestrator, at the moment of PoC development they were not directly integrated with the Slicing blocks, as depicted in Figure 36 in Subsection 5.3.1. It is because such integration is not needed if the Slicing blocks (RAN NSSMF, NFMF) are not allowed to autonomously spawn new instances of VNFs nor VLs underlying the MFIs and MLIs, which is assumed in the operational scope of the Subject Orchestrator as described above. Thus, this simplification allowed decreasing the architectural complexity of the Subject Orchestrator at the stage of development utilized by the author for PoC. However, lack of the mentioned integration **blocks the possibility of extending the PoC to support <AS3> and <AS4>**, which assume spawning new instances of the *lowest-level Runtime Slicing Entities* as a result of the design process. For this reason, <AS3> and, possibly, <AS4> are considered for implementation as a future work that requires prior improvements to be applied in the Subject Orchestrator itself.

5.3.3. Characteristics of MFs supported by the Subject Orchestrator

The Samsung's prototype has been built to operate a particular version of the Samsung Virtualized RAN (VRAN) functions: gNB-DU, gNB-CU-UP, gNB-CU-CP (see lower part of Figure 36). At the time of the prototype development, these *Slicing Entities* were also at their prototyping stage, being subjects of intensive research and development [167]. Due to this fact, the set of SLS *attributes* that were possible to be modeled as their *Capabilities* was limited[145]. In total, the utilized gNB MFs support seven SLS *attributes* that are schematically depicted in Figure 37 and briefly described in the following paragraphs.

Four attributes exhibit low-level semantics and three attributes exhibit high-level semantics. In the list below, the meaning of each *attribute* as a *Requirement* (when present in the `SliceProfile`) and as a *Capability* is explained. The four low-level SLS attributes are supported by the gNB-DU:

1. `prbPartitionMinRatioDl` - indicates the percentage of PRBs[8] that in the Cells controlled by the gNB-DU are preferentially utilized by the downlink traffic of the allocated Slice. Preferential utilization means that the PRBs are guaranteed when the assigned Slice needs them, but can be utilized by other Slices in case the demanded QoS characteristics of the assigned Slice's traffic are met.

[144] This is performed by the Operator in an assisted manner utilizing the GUI tools of the NFV-MANO and EMS blocks. Operator delivers the "Slice-agnostic" and "Slice-influenced" configurations (see Subsection 3.3.2) to MF and ML *Design-time Slicing Entities* to spawn correctly configured instances.

[145] As discussed in Subsection 2.3.2, providing support even for a single, high-level SLS attribute by a given *Slicing Entity* may require complex and long-lasting development at multiple layers of the telco stack, usually planned in long-term roadmaps. Increasing the count of high-level SLS *Requirements* supported by the Subject Orchestrator is targeted as one of the PoC goals, which is described in the upcoming Subsection 5.4.3.

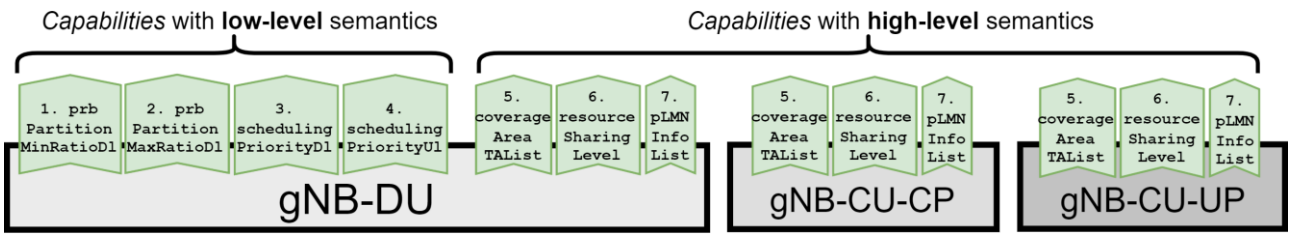


Figure 37. *Capabilities attributes* exposed by the MF *Slicing Entities* supported by the Subject Orchestrator.

- As a *Requirement*, this attribute indicates the demanded percentage of PRBs for preferential utilization by the assigned Slice.
- As the gNB-DU's *Capability*, it indicates the percentage of PRBs in the Cells controlled by the gNB-DU that are not reserved by any Slice and, as such, are available for preferential utilization by the designed Slice.

2. `prbPartitionMaxRatioDl` - indicates the demanded (*Requirement*) / offered (*Capability*) maximum non-exclusive consumption of PRBs by the downlink traffic of a particular Slice. This attribute defines the upper limit of PRBs consumption, even if there are unused PRBs.
3. `schedulingPriorityDl` - indicates the demanded (*Requirement*) / offered (*Capability*) priority of scheduling radio resources utilization in downlink with respect to other Slices operated in a given gNB-DU.
4. `schedulingPriorityUl` - indicates the demanded (*Requirement*) / offered (*Capability*) priority of scheduling radio resources utilization in uplink with respect to other Slices operated in a given gNB-DU.

The semantics of the `prbPartitionMinRatioDl` and `prbPartitionMaxRatioDl` *attributes* is aligned with the 3GPP's generic concept of Radio Resources Management (RRM) policies, embodied in the standardized `RRMPolicyRatio` attributes: `rRMPolicyMinRatio` and `rRMPolicyMaxRatio`, respectively (see Clause 4.3.36 of 5G NRM [32] for details). The relationship between the attributes enumerated above and their meaning for handling Slice-related traffic in the Cells managed by a given DU are further clarified in Table 6 and Figure 38 below, which have been elaborated based on [28] and the 5G NRM standard:

Table 6. Comparison of RAN slicing mechanisms.

Aspect	Traffic priority scheduling	PRB partitioning
Basic description	Slices share PRBs and the SLS of a particular Slice is guaranteed by increasing the scheduling priority of devices that have not reached the minimum guarantee value	Dedicated and/or preferential PRBs are assigned to the particular Slices to ensure that the utilizing UEs
Guarantee	Best effort: if Cell congested, SLS of some Slices may not be guaranteed	UEs get accurate SLS guarantee
Isolation	Sharing of PRBs between Slices provides soft isolation	Hard isolation of PRBs between Slices avoids their mutual influence and provides an isolation effect similar to that of dedicated frequency

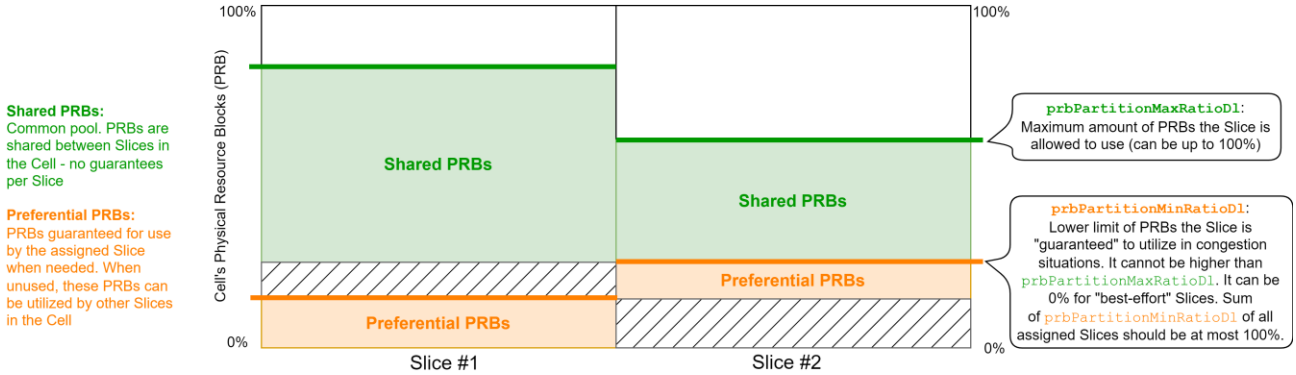


Figure 38. Physical Resource Blocks (PRB) partitioning, including the explanation of the preferential and shared utilization thresholds.

Besides the low-level SLS *attributes* described above that are exposed by gNB-DU, all gNB MFs (i.e. gNB-DU, gNB-CU-CP and gNB-CU-UP) expose 3 *attributes* with high-level semantics that directly correspond to the 3GPP-standardized SLS *attributes* defined in `SliceProfile` and `RanSubnetSliceProfile` data structures [32]. These are:

5. `coverageAreaTAList` - indicates the demanded (*Requirement*) / supported (Capability) geographical coverage area of a *Slicing Entity*, expressed as a list of TAI[61] codes.
6. `resourceSharingLevel` - indicates if a given *Slicing Entity* should (*Requirement*) / can (*Capability*) be simultaneously utilized to serve multiple Slices or it shall be dedicated for exclusive utilization to support a given Slice.

7. `pLMNInfoList` – this is an important attribute that **conveys crucial information about the supported Slice types by a given MF and the availability of their utilization by the particular Customers**. More precisely, this attribute is a list of `PLMNInfo` structures [32], each containing two sub-attributes:
   - `pLMNId` – identifies the PLMN[12], e.g. a particular MVNO utilizing the Provider's infrastructure.
   - `sNSSAI` – consists of SST and Slice Differentiator (SD) subattributes[88]. While SD meaning is not crucial in the PoC, SST[36] is an **important "umbrella" attribute** defining general characteristics of a Slice in terms of basic QoS expectancies and the nature of the supported traffic.

   By taking into account the important meaning of its sub-attributes, the semantics of the `pLMNInfoList` SLS *attribute* as a *Requirement* and *Capability* is as follows:
   - ► as a *Requirement*, it indicates for which PLMNs the Slice should be established and what basic Slice characteristics (SST) it should support. At the technical level, assignment of this *Requirement* with a given SST value to a particular MF entails configuring a range of low-level settings that optimize its functioning to the characteristic of the demanded type of Slice[146].
   - ► as a *Capability*, it indicates what Slice types (SSTs) are possible to be supported in the NSSIs allocated by the given PLMNs. For example, an MF's `pLMNInfoList` *Capability* with value:
     `[ (plmnId=X, sNSSAI=1), (plmnId=X, sNSSAI=4), (plmnId=Y, sNSSAI=1) ]`
     indicates that this MF supports Multi-Operator RAN (MORAN) feature and can be simultaneously utilized for allocating NSSIs for several PLMNs - X and Y in the example above. While PLMN X is allowed to allocate Slices of type eMBB (SST=1) and V2X (SST=4), PLMN Y is only allowed to allocate Slices of type eMBB

All aspects discussed in the previous Sections are important in the context of the improvements targeted in the PoC, which are presented in the following Section.

5.4. Shortcomings of Subject Orchestrator targeted for improvement in the PoC

In this Section, presentation of the Subject Orchestrator's technical aspects is continued. However, the discussion is now focused on its selected operational shortcomings that are targeted to be improved in the PoC. In particular, it is described how did the NSSI design, provisioning and allocation processes look like originally and what were the SLS *Requirements* originally supported in the allocation requests. Presentation of these aspects serves as a background for juxtaposition with the results of the PoC described in the upcoming Sections. It also allows for highlighting some of the operational limitations and standards-compliancy deficiencies that were possible to be eliminated in the Subject Orchestrator thanks to the application of the elaborated framework's concepts.

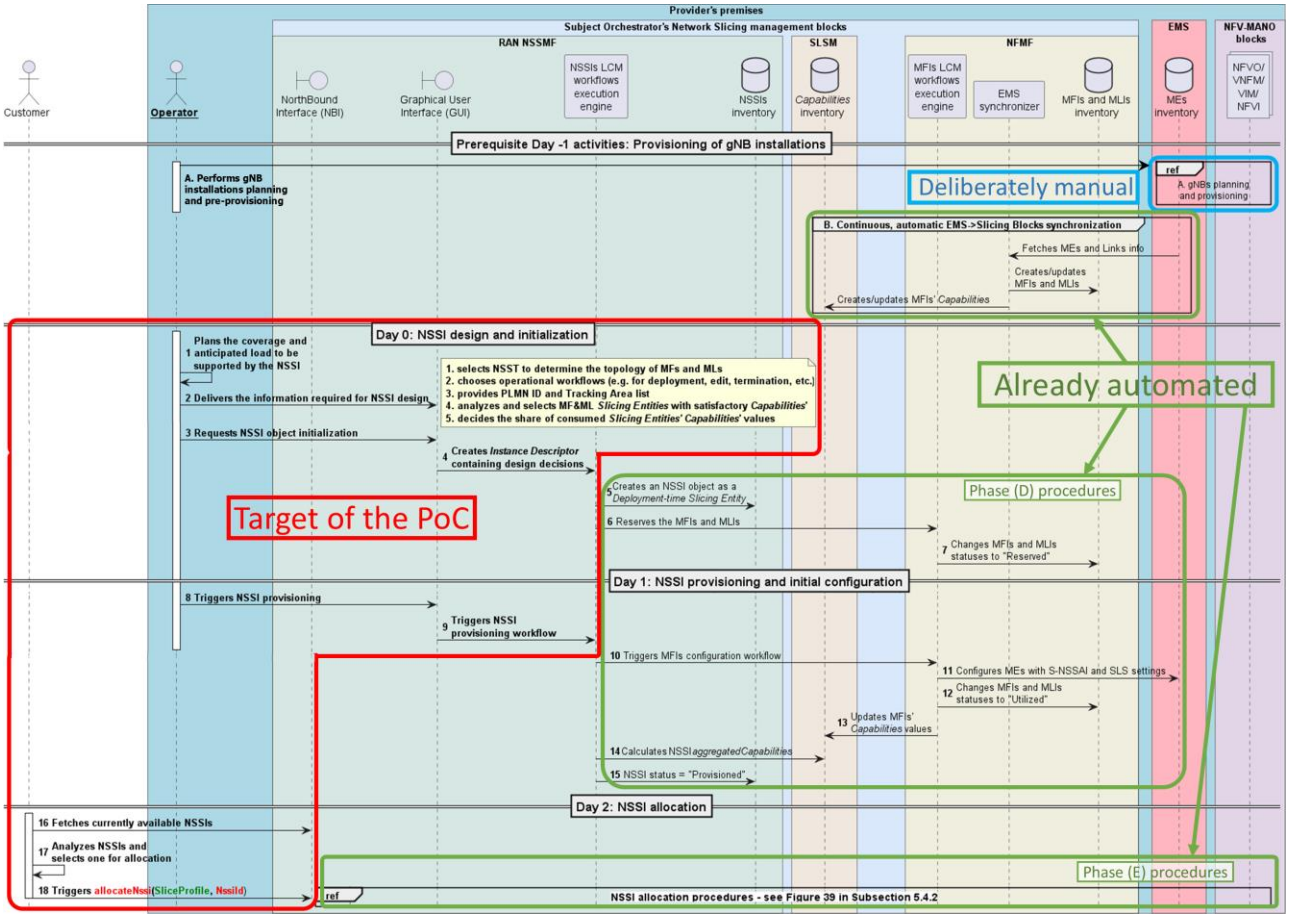
5.4.1. Shortcoming 1 - Manual NSSI design and triggering of its provisioning by the Operator

The Subject Orchestrator in its original operational scope works according to the *imperative provisioning style* principles. In particular, the process of designing a new NSSI and triggering of its provisioning in the deployment environment requires **direct involvement the human Operator**, who needs to perform a series of **manual** and **assisted** steps highlighted in Sequence Diagram 2.

As an operational prerequisite, it is assumed that the Operator provisioned the gNB installations that are designated to be utilized for Slicing (prerequisite Step A). These prerequisite activities involve: RAN sites planning, physical RUs deployment in the geographical areas to be covered, deployment and bootstrap configuration of the VNFs hosting the virtualized CU-CP, CU-UP and DUs, as well as establishment of Links between them (see lower part of Figure 36). The Operator performs these duties largely manually or in an assisted manner, utilizing the GUI tools of the NFV-MANO and EMS blocks. These prerequisite activities should stay under manual control of the Operator, as assumed in the operational scope of the Subject

[146] In particular, it configures the 5G QoS Identifier (5QI) to the particular, supported value that is mapped to the specific SST value in a given MF. The 5QI characteristics describe the packet forwarding treatment that a QoS Flow receives edge-to-edge between the UE and the UPF in terms of several performance characteristics, including the packet delay budget, packet error rate, averaging windows, maximum data burst volume, etc. See Table 5.7.4-1 [68] for the specification of the 5QI values and their assigned QoS characteristics.

Sequence Diagram 2. Original flow of NSSI design and provisioning procedures in the Subject Orchestrator, before PoC implementation.



Orchestrator (see Subsection 5.3.2) – hence, automation of these activities is not in scope of the PoC (indicated with a blue frame in Sequence Diagram 2)[147].

As soon as the gNB installations become operational, the ME objects and the Links established between them are synchronized to the Slicing blocks (prerequisite Step B). Based on the obtained information, the high-level MFI and MLI *Runtime Slicing Entities* objects are created in the NFMF's operational inventory, equipped with *Capabilities* tracked by SLSM. EMS→NFMF synchronization is performed continuously and in an automated manner (indicated with a green frame), making the MFI and MLI *Runtime Slicing Entities* ready to be considered during the design of NSSIs.

In the original operational scope, the design process (steps 1-4) and triggering of new NSSI provisioning (steps 8-9) are performed manually by the Operator, with the assistance of the GUI tools of the RAN NSSMF - **automation of these activities and exclusion of the Operator's involvement in this process is the main target of the PoC** (indicated with a red frame). The Operator begins the design process by planning the structure of an NSSI to be created (step 1). During the planning, the geographical coverage of the designed NSSI needs to be determined and the operational load to be handled by the NSSI, which will be incurred by the `SliceProfiles` incoming in the allocation requests from the Customers, needs to be anticipated by the Operator. Based on the planning, the Operator makes the design decisions and delivers them in the GUI wizard of the RAN NSSMF block (step 2). Among others, the Operator needs to:

- select a suitable NSST that specifies the desired topology of internal *Slicing Entities* (MFs and MLs) in the designed RAN NSSI;
- analyze the *Slicing Entities* available to be utilized as NSSI internals, their current operational load and exposed *Capabilities*;
- select the *Slicing Entities* candidates that cover the given geographical area and expose *Capabilities* values that will allow supporting the anticipated operational load;

[147] Automating some of these activities would be targeted when implementing <AS3>.

- decide on the share of *Capabilities* of the selected *Slicing Entities* that will be available for allocation for supporting the `SliceProfiles` anticipated to be assigned to the designed NSSI;
- configure other operational aspects of the designed NSSI (e.g. select the operational workflows that will be utilized during its runtime e.g. for its scaling, edition, termination, etc.).

Here it is worth noting that depending on the operational objectives, the Operator's design strategy may be to create few coarse-grained, large scale and large capacity NSSIs that will be capable of handling multiple `SliceProfiles` at once. Alternatively, designing many fine-grained, smaller scale NSSIs may be preferred, in order to be devoted to supporting a couple or even just a single `SliceProfile` per NSSI.

After finishing the design process, an *Instance Descriptor* capturing the enumerated design decisions is outputted from the GUI wizard and passed to the LCM workflow execution engine of the RAN NSSMF (steps 3-4). Based on the delivered *Instance Descriptor*, an automated workflow is executed (indicated with a green frame), which prepares an initial stub of the NSSI object (step 5) as a reserved, but not yet instantiated *Deployment-time Slicing Entity*, requesting internal *Slicing Entities* pre-provisioning and reservation of their resources via NFMF (steps 6-7). Once the NSSI *Deployment-time Slicing Entity* has been initialized, the Operator still needs to manually trigger (steps 8-9) the "New NSSI provisioning" workflow. This already automated workflow involves, among others:

- sending configuration commands via EMS to the MEs underlying the utilized MFI *Slicing Entities* (step 10-11), pushing their SLS-related settings, i.e. configuring the PRB fraction reserved for preferential and maximal usage by the NSSI (`prbPartitionMinRatioDl`, `prbPartitionMaxRatioDl`) and the priority of its traffic scheduling (`schedulingPriorityDl`, `schedulingPriorityUl`)[148];
- configuration of consumed share of *Capabilities* of the utilized MFI *Slicing Entities* (step 12-13);
- creation of the NSSI's *aggregated Capabilities* objects based on the lower-level *Capabilities* exposed by the utilized MFI *Slicing Entities* (step 14);
- update of the NSSI object status in the operational inventory (step 15).

Due to the fact that the "New NSSI provisioning" procedures are already automated in the original operational scope of the Subject Orchestrator (indicated by the green frames in Sequence Diagram 2), these procedures are reused "as is" in the implementation of the ASD workflow in phase (D), as it will be shown in the upcoming Subsection 5.6.1.

After step 15 is accomplished, the newly created NSSI is up and running. However, it does not have any `SliceProfiles` allocated yet. It is now, that such an already existing NSSI can be considered for satisfying the allocation requests arriving from the Customers (steps 16-18), which is handled as described in the following Subsection.

5.4.2. Shortcoming 2 - Allocation of existing NSSIs only, selected by Customer

In its original operational state, the Subject Orchestrator is capable of handling the allocation requests by assigning the input `SliceProfile` *Requirement*s **only to the already existing NSSIs**. This limitation stems directly from the lack of design activities automation, which entails that dynamic design and provisioning of new NSSIs in response to the spontaneously arriving allocation requests is not possible. As a consequence, **prior to handling the allocation requests** triggered by the Customers, **a pool of NSSIs needs to be manually created beforehand by the Operator**, by performing a series of manual and assisted steps described in the previous Subsection 5.4.1. Due to these limitations, the NSSI allocation workflow before PoC implementation was simplistic and limited in scope. Its organization with respect to the general phases (A)-(E) defined in Subsection 5.1.1 is schematically depicted in Figure 39 and briefly described in the following paragraphs.

The workflow is triggered by the Customer who invokes the `allocateNssi(SliceProfile, NssiId)` operation (badge 1 in Figure 39, step 18 in Sequence Diagram 2). In the original state of the Subject Orchestrator this operation is standards-inspired but **not compliant**. It is because lack of implementation of the design activities (in particular, the *evaluation* and *selection* activities) puts the responsibility of choosing an NSSI capable of meeting the `SliceProfile` *Requirements* on the Customer (steps 16-17 in Sequence Diagram 2),

[148] Detailed description of these SLS *attributes* has been provided in Subsection 5.3.3.

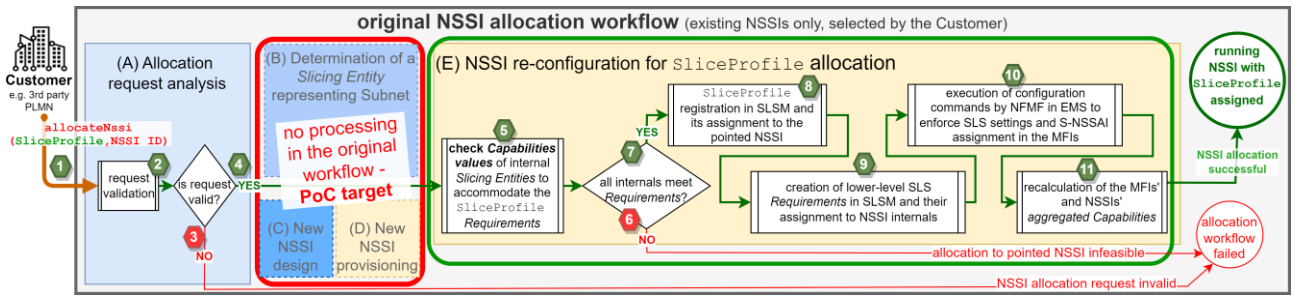


Figure 39. Activities involved in the original NSSI allocation workflow in the Subject Orchestrator, before PoC implementation.

who needs to pass the selected NSSI's ID in the invocation. Such assumption violates the signature of the `allocateNssi` operation defined in the 3GPP 28.531 [31] standard, which dictates that the implementation of this operation should accept solely a `SliceProfile`. This deficiency, among others, is targeted to be fixed as a result of the PoC.

In the (A) "Allocation request analysis" phase, the allocation request is validated (badge 2). If the NSSI pointed in the request does not exist, or the input `SliceProfile` contains unsupported SLS *attributes* or unsupported values, the allocation is unsuccessful (badge 3). Otherwise, the workflow proceeds directly to phase (E) "NSSI re-configuration for `SliceProfile` allocation" (badge 4). Importantly, in the original allocation workflow there is **no processing in the (B), (C) and (D) phases** (indicated with a red oval). It is because the NSSI design and provisioning had to be performed manually by the Operator (as described in the previous Subsection 5.4.1), before the reception of the `allocateNssi` requests became possible.

In phase (E) "NSSI re-configuration for `SliceProfile` allocation", the internal *Slicing Entities* of the pointed NSSI are firstly checked for feasibility of accepting new *Requirements* (badge 5). If the internals refuse because of their currently insufficient *Capabilities* values, the allocation is unsuccessful (badge 6). Otherwise, the workflow proceeds with the NSSI re-configuration procedures (badges 7-11) that are already automated in the original scope of the Subject Orchestrator (indicated with green oval). As such, they are reused in the PoC end-to-end NSSI allocation workflow (this is shown in the upcoming Subsections 5.6.4 and 5.6.5).

The originally implemented **`allocateNssi(SliceProfile, NssiId)`** operation with a simplistic workflow described above is not only limited in scope, as only allocation to the already existing NSSI is permitted, but it also **does not follow the *declarative provisioning style***. It is because it requires the Customer to firstly acquire detailed knowledge about the NSSIs currently operated in the Subject Orchestrator, analyze them and thoughtfully choose one of them to be used for `SliceProfile` allocation. The implementation of the ASD PoC aims at removing these deficiencies, enabling full alignment of the `allocateNssi` operation's signature and behavior to the standardized, declarative perception.

5.4.3. Shortcoming 3 - Limited set of supported high-level SLS attributes

By analyzing the list of *Capabilities* exposed by the MFs that are managed by the Subject Orchestrator (see Subsection 5.3.3), it can be noted that they **do not explicitly support** many of the desirable **high-level SLS attributes** defined in the 3GPP standards, e.g. `dLLatency`, `uLLatency`, `dLThptPerSlice`, `uLThptPerSlice`, `maxNumberofUEs` and others. It is because the delivery of such high-level guarantees by the individual gNB MFs (and, consequently, the NSSIs containing them) can only be credibly achieved via constant, intelligent closed-loop adaptation of the radio resources utilization, e.g. by continuous manipulation of the PRBs and scheduling priorities assigned to the managed NSSIs, taking into account their current and predicted load. Such advanced, dynamic management of the radio resources is targeted by the Open RAN (O-RAN) non-Real Time (non-RT) and near-RT Radio Intelligent Controller (RIC) [168] components, which unfortunately were not in scope of the Subject Orchestrator at its prototyping stage.

Due to this limitation, the NSSI objects managed by the Subject Orchestrator instead of exposing *Capabilities* with the mentioned high-level semantics, are only aggregating the MFs' *Capabilities* values and re-exposing them, as schematically depicted in Figure 40 below. As a result, the NSSI objects expose the same set of *Capabilities* as the orchestrated MFIs, exhibiting a mix of low and high-level semantics. Consequently, in the

`allocateNssi(SliceProfile)` invocation, the Customers may only specify SLS *Requirements* against the limited set of seven *Capabilities* attributes presented in Subsection 5.3.3.

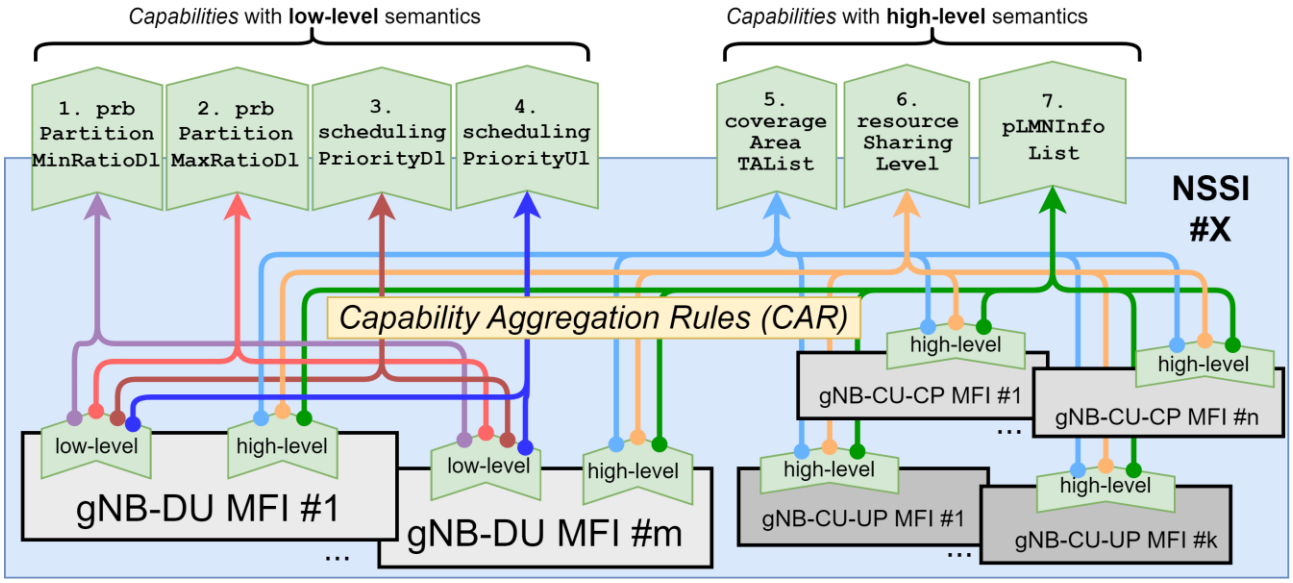


Figure 40. *Aggregated Capabilities* exposed by the NSSI objects supported by the Subject Orchestrator.

Although support of the aforementioned high-level SLS *attributes* is not obligatory, it would benefit the Customers, enabling expression of their demands in an abstract, QoS-oriented way, instead of specifying *Requirement*s in terms of low-level technical properties of RAN equipment operation (PRBs, scheduling priorities). Implementation of the ASD PoC aims at improving this aspect by increasing the count of high-level SLS *attributes* supported in the input `SliceProfile`. Despite the lack of their native support[149], their handling is possible, under certain assumptions, by leveraging the automation framework's concepts, as it will be described in the upcoming Subsection 5.6.2.

5.4.4. Summary of shortcomings targeted in the PoC

In summary, the Subject Orchestrator in its original operational scope exhibits three major shortcomings related to NSSIs design and allocation procedures that are targeted to be improved by implementing the ASD PoC:

1. Only **manual/assisted NSSI design and provisioning** is supported:
   - × This shortcoming requires the Operator to manually plan, design and provision the appropriate NSSIs before any NSSI allocation requests from the Customers can be handled.
   - ► Implementation of the ASD PoC shall relief the Operator from this duty, enabling fully-automated, dynamic design and provisioning of NSSIs suited to fulfill the incoming demands.
2. Due to the lack of design automation, allocation is possible **only to the already existing NSSIs**. Moreover, the Customer invoking the `allocateNssi(SliceProfile, NssiId)` operation **is required to point the NSSI to which the `SliceProfile` shall be assigned:**
   - × This shortcoming requires the Customer to possess detailed knowledge about the existing NSSIs and thoughtfully decide which one to utilize for allocation. Such approach does not follow the *declarative provisioning style* and is not standards-compliant.
     - ► Application of the ASD PoC should remove the constraint of passing the NSSI ID in the `allocateNssi` operation, making its signature fully standards-compliant. As a result, the Customer shall be relieved from possessing any knowledge about the NSSIs managed by the Subject Orchestrator and any intelligence of their choice, making the utilization of the `allocateNssi` operation fully *declarative*.
   - × Such an implementation of the `allocateNssi` operation limits the dynamicity of NSSIs management, requiring the Operator to perform a manual pre-provisioning of a pool of NSSIs that will be capable

[149] Here it shall be underlined that handling of the mentioned, high-level SLS *attributes* by the automation framework's mechanisms is seamless once they become natively supported in the utilized *Slicing Entities* or by the Subject Orchestrator (e.g. implementation of the O-RAN non-RT RIC [168]).

of supporting the anticipated load of incoming `SliceProfiles`. Such functioning of the Subject Orchestrator is only sufficient for managing NSSIs in a "static slicing" manner (see Subsection 2.3.6).

- ► Application of the ASD PoC shall enable automatic spawning of new NSSIs if the existing ones do not satisfy the incoming `SliceProfile` SLS *Requirements*. Such feature will increase the dynamicity of management performed by the Subject Orchestrator, paving the way towards NSSIs orchestration in the "dynamic slicing" manner[150].

3. A **limited subset of high-level SLS *Requirements*** standardized by 3GPP is supported:
   - × It is due to the lack of explicit support for the high-level SLS *Capabilities* in the gNB MFs and limited advancement of the SLS assurance functionalities implemented in the Subject Orchestrator.
   - ► Implementation of the ASD PoC should showcase the possibility of supporting high-level SLS *Requirement*s, leveraging the organization of automation framework elements in a variant that will enable such support.

Having in mind the ultimate goals of the PoC specified in Section 5.1 and the targeted characteristics of the Subject Orchestrator presented above, the PoC implementation details can now be presented.

5.5. Implementation of the individual framework's elements

In this Section, selected implementation details of the PoC are described. Firstly, in Subsection 5.5.1, the placement of the ASD PoC elements within the Subject Orchestrator's architecture is presented. In the next Subsections, the object-oriented decomposition of the PoC solution is discussed. In Subsection 5.5.2, the implemented models are detailed, emphasizing which modeling decisions were crucial for the enablement of automated processing in the individual design activities. The implementation of the design activities themselves is described in Subsection 5.5.3.

It shall be noted that while the current Section is focused on discussing the PoC's **elementary building blocks** (individual models and methods), it is the upcoming Section 5.6 that discusses the details of **their composition** into an end-to-end automated NSSI allocation workflow.

5.5.1. ASD PoC elements in the Subject Orchestrator's architecture

Figure 41 is similar to Figure 36 in Subsection 5.3.1, but now it presents the Subject Orchestrator's architecture after ASD PoC implementation. The PoC involves elements that have been either newly implemented (marked in yellow), reused "as is" from the Subject Orchestrator's original state (marked in white) or enhanced (marked in blue). It is because the Subject Orchestrator in its original state, although operationally limited as described in the previous Sections, already included the implementation of several crucial concepts of the automation framework elaborated in this thesis. This is, among others, thanks to the contributions of the author, who strived for maximal architectural and implementation-level alignment of the developed Subject Orchestrator to the concepts presented in this thesis.

The ASD PoC elements are incorporated in the (1) Network Slicing architectural blocks of the Subject Orchestrator: RAN NSSMF, NFMF and SLSM. The RAN NSSMF block has been selected to contain the core ASD logic. Another architectural option would be to implement the ASD logic inside of a separated, dedicated component to which the NSSI design request would be forwarded. Such an externalized component could potentially support multiple Subject Orchestrators, performing a centralized design process. However, implementation of such an architectural variant is not essential for the verification of the elaborated concepts and, as such, is considered as a future work discussed in Subsection 5.7.3.

The PoC elements that in Figure 41 are depicted in a high-level manner are discussed in detail in the following Subsections.

[150] As explained in Subsection 4.7.2, fully "dynamic slicing" orchestration is enabled by implementing <AS3>, which is not targeted in the PoC.

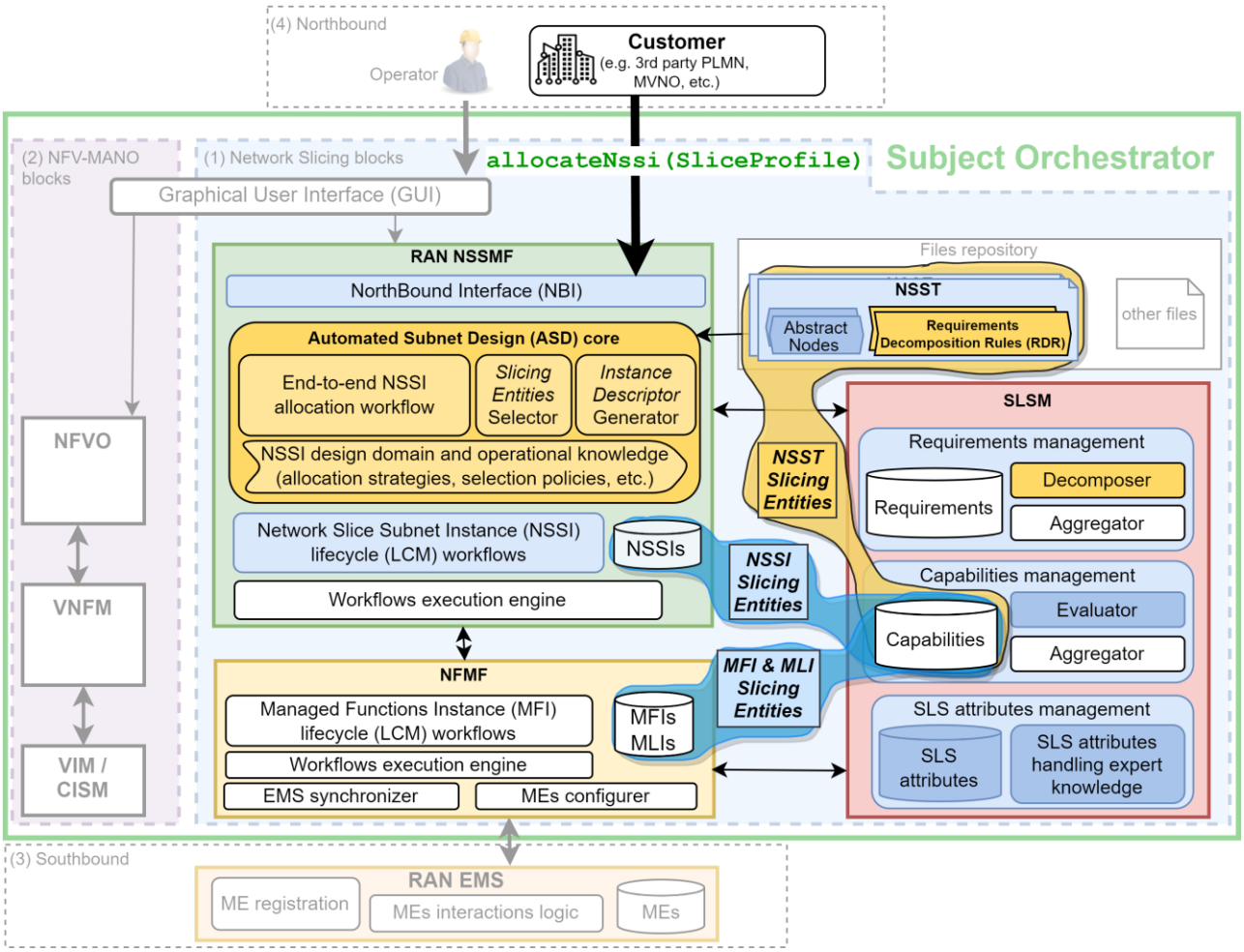


Figure 41. Subject Orchestrator's architecture after ASD PoC implementation. The elements comprising the PoC are distinguished with colors: newly implemented (yellow), enhanced (blue) and reused (white) from original Subject Orchestrator's implementation.

5.5.2. Models' implementation

The ASD PoC comprises the implementation for a range of models and activities of the conceptual framework that allow embodying the assumed scope of the PoC defined in Subsection 5.1.2. The object-oriented modeling of the most important PoC elements is depicted in Figure 42 leveraging the Unified Modeling Language (UML) class and component diagrams combined in a single picture.

To facilitate the analysis, the relevant parts of the diagram are magnified in the upcoming Subsections. Moreover, due to a large number of introduced models, the intermediary Subsection 5.5.2.4 presents a sample arrangement of instances of the presented models during Subject Orchestrator's runtime, which allows to clarify and consolidate the understanding of their operational semantics.

It must be underlined that the **modeling approach** employed in the construction of the PoC models, which follows the concepts of the elaborated framework, constitutes the **key enabler for achieving automation in the design activities**. In particular, the domain knowledge conveyed in the implemented models constitutes an essential input that steers the execution of logic in the individual design activities and the entire end-to-end allocation workflow. Thus, careful study of the models' construction described in the Subsections below, **at least the bolded parts**, is important for a proper understanding of the working mechanisms employed in the implementation of the automated design activities, which are described in Section 5.5.3 that follows.

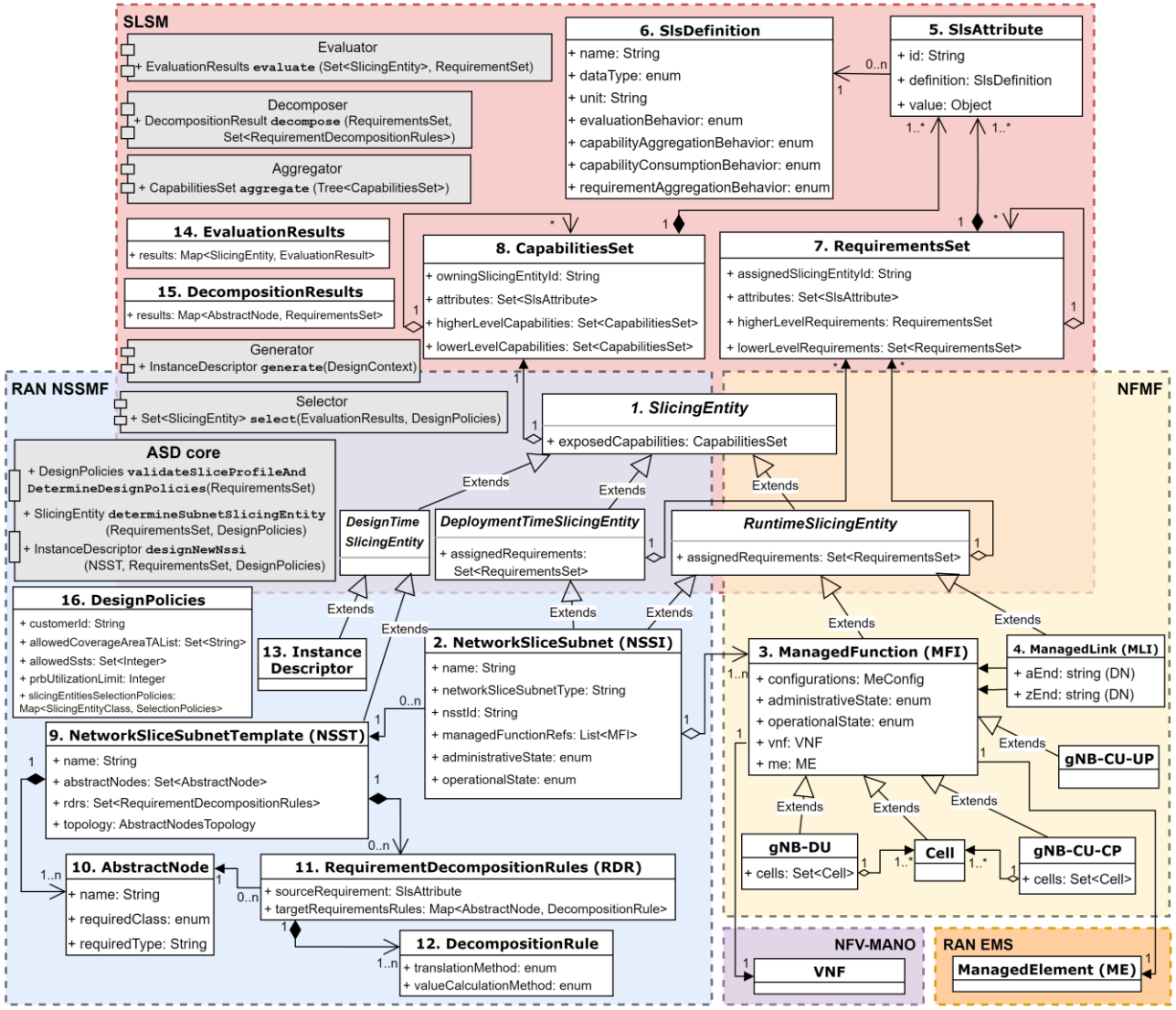


Figure 42. UML diagram of the PoC elements incorporated inside of the Subject Orchestrator's components: RAN NSSMF (blue background), NFMF (yellow background) and SLSM (red background). The implemented models are indicated with white color, whereas the functional modules are depicted in grey.

5.5.2.1. Models of abstract Slicing Entities and their concretizations

NOTE: The models' numbers correspond to the numbering utilized in Figure 42.

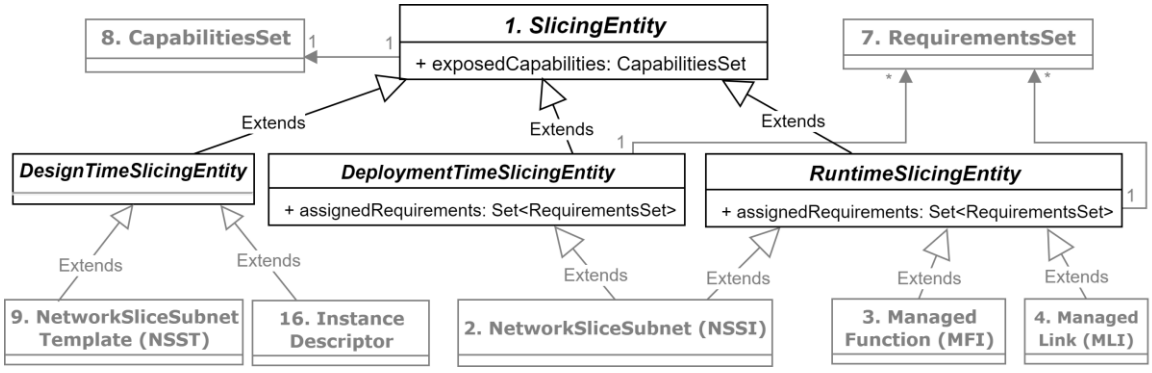


Figure 43. *Slicing Entities* abstract and concretized models supported in the PoC, as well as their relationship to SLS management models.

1. ***`SlicingEntities`*** abstract models and their concretizations:

`SlicingEntity` is an abstract model of a building block out of which a Subnet may be designed. Each `SlicingEntity` exposes a `CapabilitiesSet` (see model no. 8) - **this modeling decision is crucial for enabling unified and generic processing of *Slicing Entities* in the design activities, abstracting away any of their vendor or implementation-specific particularities**. The `SlicingEntities` models are organized into a

hierarchy of abstract models comprising: *SlicingEntity*, *DesignTimeSlicingEntity*, *DeploymentTimeSlicingEntity* and *RuntimeSlicingEntity*. In the ASD PoC, the following concretizations of the abstract *SlicingEntities* models are implemented:

- *NSSI Deployment-time* & *Runtime Slicing Entity* - modeled as a combination of an NSSI object (see model no. 2) managed in RAN NSSMF with a CapabilitiesSet object managed in SLSM: *NSSI Slicing Entity* = NSSI + CapabilitiesSet;
- *MFI Runtime Slicing Entity* - modeled as an MFI object (see model no. 3) managed in NFMF combined with a CapabilitiesSet managed in SLSM: *MFI Slicing Entity* = MFI + CapabilitiesSet;
- *MLI Runtime Slicing Entity* is modeled as an MLI object managed in NFMF, with an empty CapabilitiesSet (see model no. 4) ;
- *NSST Design-time Slicing Entity* - modeled as an NSST object (see model no. 9) in the files' repository combined with a CapabilitiesSet managed in SLSM: *NSST Slicing Entity* = NSST + CapabilitiesSet;
- *InstanceDescriptor Design-time Slicing Entity* - modeled as the InstanceDescriptor object (see model no. 13) where the CapabilitiesSet is not persistently stored, but kept shortly in-memory because it is immediately transferred to the SLSM inventory after NSSI instantiation. *InstanceDescriptors* are used solely as deposits of the design decisions made during the design process and are not considered as *candidates* in the subsequent executions of the design process – see model no. 13 for details.

Note: as depicted in the UML diagram in Figure 43, *RuntimeSlicingEntities* (NSSIs, MFIs) and *DeploymentTimeSlicingEntities* (NSSIs) can have operationally assigned RequirementsSets (see model no. 7), whereas *DesignTimeSlicingEntities* (NSSTs, InstanceDescriptors) cannot.

5.5.2.2. Models of Deployment-time and Runtime Slicing Entities

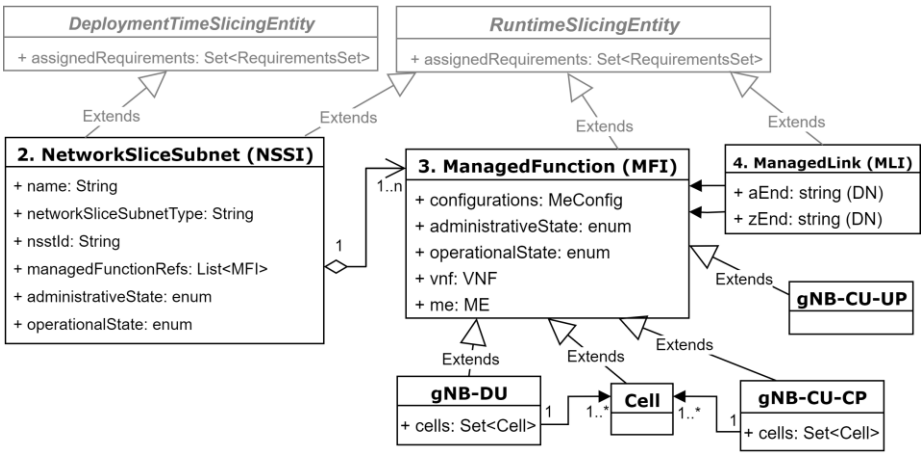


Figure 44. Concretizations of the abstract *Deployment-time* and *Runtime Slicing Entities* models supported in the PoC.

2. **NetworkSliceSubnet** (**NSSI**) model:

NSSI model is utilized for representing operational Subnet objects as combined *Deployment-time* & *Runtime Slicing Entities*[151]. NSSI model provides the attributes assumed by the NetworkSliceSubnet model in the 3GPP specifications [32], among others: the networkSliceSubnetType (always "RAN" in the PoC), its administrativeState, operationalState, the list of internal MFIs (managedFunctionRef) and SliceProfiles assigned to it (sliceProfileList).

3. **ManagedFunction** (**MFI**) set of models:

A given ManagedFunction is modeled as an abstract, combined representation of a ManagedElement (ME) object managed in EMS and the VNF object managed in NFV-MANO. There are specific models for the gNB-DU, gNB-CU-CP and gNB-CU-CP MFIs which encapsulate all the Slicing-related interactions with the Samsung's gNB MEs and the underlying Cell objects managed from the gNB-CU-CP and gNB-DU (see lower right part of Figure 42).

[151] NSSI as a *Deployment-time* or *Runtime Slicing Entity* is indicated by the value of the operationalState attribute: RESERVED or RUNNING.

4. Simplified **`ManagedLink`** (**`MLI`**) model:

`MLI` is modeled by utilizing the information from EMS about the operationally established `Links` between the `MFIs`. Exposition of *Capabilities* by `MLIs` is out of scope in the Subject Orchestrator (such extension is considered as a future work, see Subsection 5.7.3). For this reason, `MLIs` in the design process are not evaluated for SLS *Requirements* fulfillment, but only for validating the expected topology of Links between the `MFIs` in the designed `NSSI` (details provided in the upcoming Subsection 5.6.5).

5.5.2.3. Models for management of SLS Requirements and Capabilities

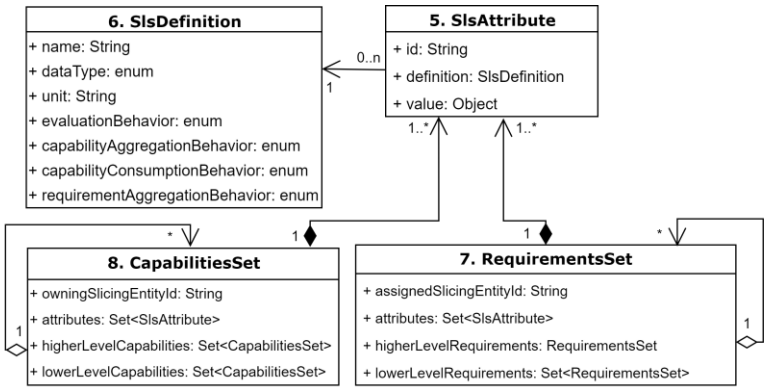


Figure 45. SLS *Requirements* and *Capabilities* management models.

5. **`SlsAttribute`** model:

`SlsAttribute` model is defined in a generic manner - depending on the context, its instance represents either a *Requirement* (if present in a `RequirementsSet`, see model no. 7) or a *Capability* (if present in a `CapabilitiesSet`, see model no. 8). During the design process, each `SlsAttribute` **is processed in an automated manner** thanks to the definition of **behaviors** specified in its corresponding `SlsDefinition` (see model no. 6). Behaviors serve as the implementation-level realization of the concept of *Capability*/*Requirement attribute* processing methods that were discussed in Subsection 3.4.

6. **`SlsDefinition`** model:

`SlsDefinition` conveys meta information describing the `SlsAttributes` instances, including the attribute's name, data type and unit. Most importantly, this model conveys the essential **domain knowledge** about the **methods of processing** of the described `SlsAttribute` both when representing a *Requirement* or a *Capability*. In particular, specifications of **the following behaviors are crucial for enabling automated processing of `SlsAttributes` during the design activities**:

a) **`evaluationBehavior`** - specifies what relation needs to be checked to assess *Requirement* fulfillment by a *Capability*, e.g. the *Requirement* value must be lower/higher/equal to the *Capability* value, or the *Requirement* value must be within the *Capability* value represented as a set, etc.
   ➔ **Specification of this behavior is essential for performing automated *comparing* and *matching* in the `evaluate` method** described in the upcoming Subsection 5.5.3.1.

b) **`capabilityAggregationBehavior`** - specifies how the *Capability* value of a higher-level *Slicing Entity* is calculated from the *Capabilities* of lower-level *Slicing Entities*, e.g. min, max, sum, subtraction, set sum, etc.
   ➔ **Specification of this behavior is essential for performing automated calculations in the `aggregate` method** described in the upcoming Subsection 5.5.3.4.

`SlsDefinition` model is not only essential for automating the design process but it also serves as a **key input for the *Requirements*&*Capabilities* management mechanisms of the SLSM component**, executed continuously during the Subject Orchestrator's operation. In particular, the definitions of the following behaviors are essential for assuring that the *Slicing Entities* utilized in the design process have properly calculated values of their *Capabilities*, taking into account the *Requirements* assigned to them (see Subsection 5.5.2.4 for a clarifying example):

c) `capabilityConsumptionBehavior` - specifies how the *Requirement* value impacts ("consumes") the *Capability* value if it gets assigned to the owning *Slicing Entity*, e.g. subtraction, set difference, no impact, etc.

d) `requirementAccumulationBehavior` - specifies how an accumulated value of a *Requirement* is calculated in case multiple `RequirementsSets` are assigned to a single *Slicing Entity*, e.g. min, max, sum, set sum, etc.

Below, two examples of `SlsDefinition` model instances are presented for clarification:

I. `SlsDefinition` for the "`coverageAreaTAList`" `SlsAttribute` is as follows:
- `dataType = Set<String>`
- `unit` = none - elements of the set represent TAI[61] codes

a) `evaluationBehavior = SET_CONTAINS`
   - in order for the `coverageAreaTAList` *Capability* (e.g. of an NSSI) to be evaluated as fulfilling the *Requirement* (e.g. in a `SliceProfile`), the set of TAI codes contained in the *Capability* must contain all the TAI codes defined in the *Requirement*

b) `capabilityAggregationBehavior = SET_INTERSECTION`
   - when the `coverageAreaTAList` *Capability* of a *higher-level Slicing Entity* (e.g. of an NSSI) is determined, its value is calculated as an intersection of `coverageAreaTAList` *Capabilities* of the underlying *Slicing Entities* (NSSI internals)

c) `capabilityConsumptionBehavior = NO_IMPACT`
   - this means that the *Capability* value is not influenced ("not consumed") by the assignment of a *Requirement* – it is because the `coverageAreaTAList` of a given *Slicing Entity* does not change when a *Requirement* is assigned to it

d) `requirementAccumulationBehavior = SET_SUM`
   - this means that if there are multiple `RequirementsSets` assigned to a given *Slicing Entity*, an aggregated, single value of the `coverageAreaTAList` *Requirement* posed against the `coverageAreaTAList` *Capability* of that *Slicing Entity* constitutes all the TAI codes contained in all `RequirementsSets` assigned to it

II. `SlsDefinition` for the "`prbPartitionMinRatioDl`" `SlsAttribute` is as follows:
- `dataType = Integer`
- `unit`: "%"

a) `evaluationBehavior = HIGHER_OR_EQUAL`

b) `capabilityAggregationBehavior = MIN`

c) `capabilityConsumptionBehavior = SUBTRACTION`

d) `requirementAccumulationBehavior = SUM`

The ASD PoC reuses `SlsDefinitions` for the 7 `SlsAttributes` that were originally supported in the Subject Orchestrator (described in Subsection 5.4.3): `coverageAreaTAList`, `prbPartitionMinRatioDl`, `prbPartitionMaxRatioDl`, `schedulingPriorityDl`, `schedulingPriorityUl`, `pLMNInfoList`, `resourceSharingLevel`. Additionally, in order to showcase the possible support for high-level SLS *Requirements*, `SlsDefinitions` for 5 high-level SLS *attributes* defined in the 3GPP standardized `RANSliceSubnetProfile` data structure [32] have been added: `dLLatency`, `uLLatency`, `dLThptPerSlice`, `uLThptPerSlice`, `maxNumberofUEs`.

It is worth underlining that the genericity of the utilized **modeling approach, leveraging generic definition of behaviors, allows new `SlsDefinitions` to be easily added** when they are natively supported by the *Slicing Entities* utilized in the design process.

7. **`RequirementsSet`** model:

A given `RequirementsSet` is a grouping of `SlsAttributes` that represent *Requirements* against a *Slicing Entity*. `RequirementsSet` model is generic – it can be used to represent *Requirements* against *Slicing Entities* at any level in the hierarchy, e.g. `SliceProfiles` posed against NSSIs, MF-level *Requirements*, Cell-level *Requirements*, etc. Among others, **`RequirementsSet` model serves as the implementation level representation of `SliceProfiles`**.

`RequirementsSets` can be organized into hierarchies, i.e. higher-level `RequirementsSet` maps to multiple lower-level `RequirementsSets`. For example, `RequirementsSet` against an NSSI (`SliceProfile`) is decomposed into `RequirementsSets` against internal MFIs. `RequirementsSets` can be assigned to the

operationally existing *Slicing Entities* (*Deployment-time* or *Runtime*) or may not have an owner yet (e.g. `SliceProfile` after allocation request reception and during the design process).

8. **`CapabilitiesSet`** model:

   `CapabilitiesSet` is owned by a single *Slicing Entity* (see the 1 to 1 relationship in Figure 43). `CapabilitiesSet` internally contains individual `SlsAttributes` (see model no. 6) that represent selected characteristics of a *Slicing Entity*, allowing their assessment against the *Requirements* posed against it during the design process. `CapabilitiesSet` model is generic - it can be used to represent the characteristics of *Slicing Entities* of any type, in any lifecycle stage and at any abstraction level. `CapabilitiesSets` can be organized into hierarchies, i.e. higher-level `CapabilitiesSet` derives from multiple sets of lower-level ones, e.g. NSSI's `CapabilitiesSet` derives from multiple `CapabilitiesSets` of its internal MFIs.

   In the current state of the Subject Orchestrator, management of `CapabilitiesSets` by SLSM is performed for the NSSI and MFI *Runtime Slicing Entities* as well as for the NSST *Deployment-time Slicing Entities*. MLI *Capabilities* are not supported in the PoC (see model no. 4). `CapabilitiesSets` of `InstanceDescriptors` are also not managed in SLSM, but are stored ephemerally in memory (see model no. 13) as they are immediately utilized for instantiating the designed NSSI.

5.5.2.4. Organization of SLS management objects in a sample scenario

To clarify the characteristics of the models introduced in the previous Subsections, Figure 46 depicts the logical organization of their instances during the Subject Orchestrator's operation in a sample scenario. This simple scenario assumes existence of two NSSIs managed in RAN NSSMF: NSSI #A and NSSI #B. Internally, both NSSIs utilize MFIs managed in NFMF: shared gNB-CU-CP#1 and gNB-DU#1 instances, as well as dedicated gNB-CU-UP#1 and #2 instances. The gNB-CU-CP#1 and gNB-DU#1 manage Cells covered by the gNB-RU connected to the gNB-DU#1 (RU is omitted in the diagram). Besides the enumerated NSSI and MFI *Runtime Slicing Entities*, there is also the *NSST Design-time Slicing Entity* based on which NSSI#A and NSSI #B have been designed.

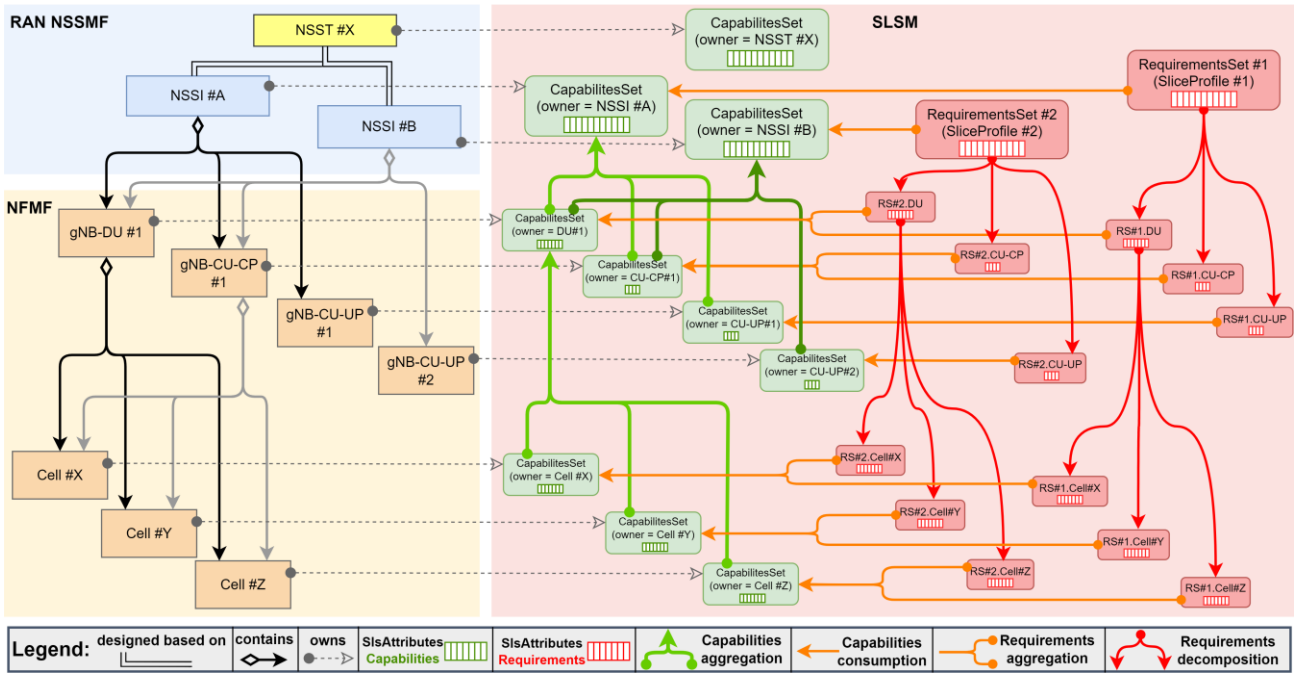


Figure 46. Organization of *Slicing Entities*, SLS *Requirements* and *Capabilities* objects in a sample scenario, including graphical presentation of *Requirements*/*Capabilities attributes* transformation methods.

Each of the enumerated *Slicing Entities* exposes a `CapabilitiesSet` managed inside of the SLSM component. Depending on the *Slicing Entity*'s class and type, their corresponding `CapabilitiesSets` contain a particular set of `SlsAttributes` (see Subsection 5.4.3), visualized in Figure 46 as green rows of tiny rectangles. The `CapabilitiesSets` of the *lower-level Runtime Slicing Entities* are aggregated towards the `CapabilitiesSets`

of the *higher-level Runtime Slicing Entities*, i.e. Cells → DU or DU, CU-CP, CU-UP → NSSI. The `CapabilitiesSets` of NSSTs are defined statically `SlsAttributes` with ranges of supported values.

In the presented scenario, both NSSI #A and NSSI #B are assigned with one `SliceProfile` each (represented as `RequirementsSets` #1 and #2). These high-level `RequirementsSets` are reflected on the lower-levels by the decomposed MF-level and Cell-level `RequirementsSets`, which are assigned to the `CapabilitiesSets` of the targeted *Slicing Entities*. Each of the abovementioned `RequirementsSets` contains a set of `SlsAttributes` (visualized in Figure 46 as rows of red tiny rectangles), that are posed against the `SlsAttributes` inside of the `CapabilitiesSets` exposed by the corresponding *Slicing Entity*. As such, these `RequirementsSets` consume a given share of values of the *Capabilities attributes* of *Slicing Entities* to which they are assigned.

During the Subject Orchestrator's operation, the SLSM component continuously tracks the `RequirementsSets` assigned to the managed *Runtime Slicing Entities* and **determines the current state of their `CapabilitiesSets`**. More precisely, SLSM calculates the `SlsAttributes` values contained in the `CapabilitiesSets` according to the behaviors specified in their `SlsDefinitions`: `capabilityAggregationBehavior`, `capabilityConsumptionBehavior` and `requirementAccumulationBehavior`. By performing these calculations, SLSM is able to verify `RequirementsSets` fulfillment by the *Runtime Slicing Entities*' `CapabilitiesSets` and determine the remaining values of their *Capabilities* that are still available for consumption. Due to the fact that NSSTs are static and cannot be assigned with `RequirementsSets` (as NSSTs are `DesignTimeSlicingEntities`) their `CapabilitiesSet` does not change during Subject Orchestrator's runtime.

Here is shall be noted that the **SLSM's continuous operation as described above is crucial for assuring that the values of *Slicing Entities Capabilities* are credibly calculated and exposed for consideration in the design process** – meeting this condition is an essential factor for organizing an automated design process, which was stated in Subsections 3.1.2 and 3.1.3.

Besides serving as a clarification of the models' construction and semantics, the descriptions above also allow realizing how many operational objects related to SLS *Requirements*/*Capabilities* management are maintained by the Subject Orchestrator during operation, even in a simple scenario. As depicted in Figure 46, assignment of a single `SliceProfile` to a given NSSI entails creation of a `RequirementsSet` object not only for that NSSI, but also for each *Slicing Entity* comprising the NSSI internally. This contributes to the **increased computational load** while performing *Capabilities aggregation* and other calculations by the SLSM component. As such, the count of the instances of the enumerated models at scale (i.e. the *Slicing Entities* managed in RAN NSSMF and NFMF as well as `CapabilitiesSets`, `RequirementsSets` and `SlsAttributes` managed in SLSM) determines the Subject Orchestrator's **operational load**[152]. The factors of operational load influence not only the performance of the Subject Orchestrator in its runtime, but also the performance of the design activities implemented in the ASD PoC. Hence, these important indicators are utilized for the performance evaluation of the PoC, which is discussed in Section 6.4.

5.5.2.5. Models of Design-time Slicing Entities

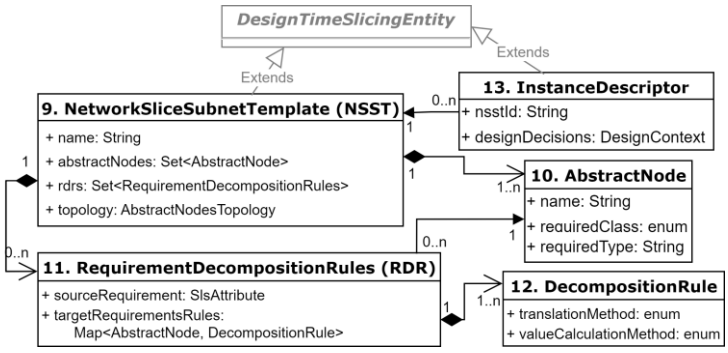


Figure 47. Models of *Design-time Slicing Entities* supported in the PoC, including models representing the abstract structure of an NSST and its *Requirements Decomposition Rules* (RDR).

[152] The operational load of the Subject Orchestrator described herein should be clearly distinguished from the load of the NSSIs' internal elements (MFIs, MLIs) in terms of handling user-plane traffic. In the given circumstances, Subject Orchestrator may handle a large count of NSSIs with large count of assigned `SliceProfiles` (which in turn entails large count of `RequirementsSets` assigned to its internals), meaning that the Subject Orchestrator is loaded. However, at the same time, the internals of the orchestrated NSSIs may be not loaded by user plane traffic.

9. **`NetworkSliceSubnetTemplate` (`NSST`)** model and its internal **`AbstractNodes`**:

`NSST` serves as an abstract blueprint for creating similarly structured, dynamically contextualized design-time descriptions of NSSIs - `InstanceDescriptors` (see model no. 13). The internal elements of an `NSST` are defined in an abstract way utilizing **`AbstractNodes`** (see model no. 10). `NSST` contains a topology of `AbstractNodes`, which represent internal *Slicing Entities* of different classes: nested Subnets, MFs and MLs. Once a `RequirementsSet` (e.g. a `SliceProfile`) is posed against the `CapabiliesSet` of an `NSST`, the values of the particular *Requirements attributes* are assigned towards the `AbstractNodes` by means of `RequirementDecompositionRules` (see model no. 11). An example of `NSST` called "`LOW_LEVEL_RAN_NSST`" that describes a typical arrangement of MFs and MLs in a RAN Subnet is presented in Figure 48 below:

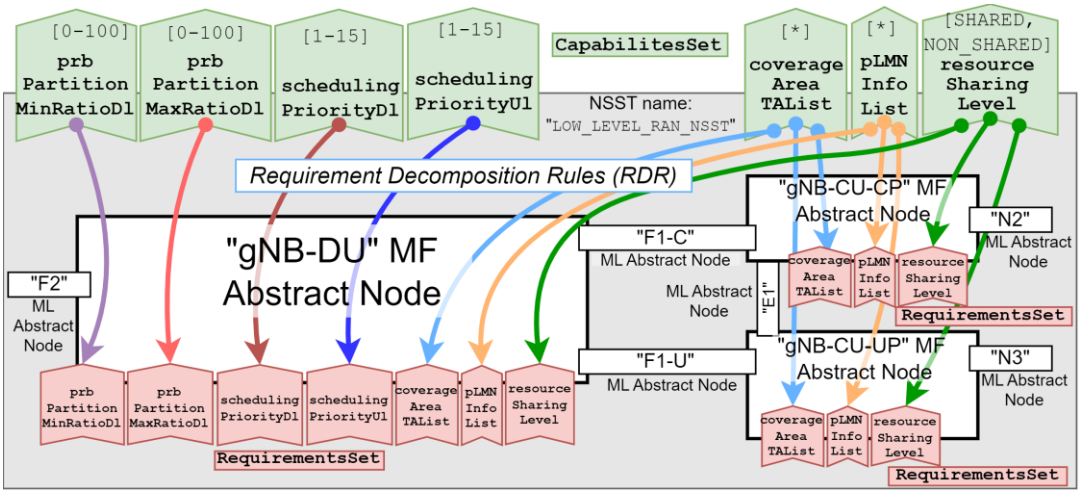


Figure 48. Organization of the `LOW_LEVEL_RAN_NSST`, including exposed *Capabilities*, *Abstract Nodes* topology and RDRs between them.

In the top, the `SlsAttributes` comprising the `NSST`'s `CapabilitiesSet` are depicted. Internally, there are 3 MF `AbstractNodes`, which constitute abstract representations of the gNB-DU, gNB-CU-CP and gNB-CU-UP MFs, as well as 6 ML `AbstractNodes` representing the F2, F1-C, F1-U, E1, N2 and N3 MLs. Between the `CapabilitiesSet` of the `NSST` and the `RequirementsSets` of the `AbstractNodes`, several `RDRs` are defined, marked by lines with different colors. The `SlsAttributes` contained in the `NSST`'s `CapabilitiesSet` serve as a blueprint for the `CapabilitiesSet` that will be exposed by an NSSI designed based on this `NSST`. To represent a particular `NSST` instance, a TOSCA-notation text file is utilized. `NSST` files are stored in the files' repository of the Subject Orchestrator.

10. **`AbstractNode`** model (utilized inside of the `NSST`):

`AbstractNode` is an abstract representation of a *Slicing Entity* out of which a Subnet shall be built internally: nested Subnet[153], MF or ML. `AbstractNode` does not point to a concretized *Slicing Entity* in the deployment environment. Instead, `AbstractNode` specifies only the expected class and type of a *Slicing Entity* (e.g. MF of type gNB-DU) as well as a `RequirementsSet` that need to be met by the *candidates Slicing Entities* considered in the design process to be mapped to a given `AbstractNode`. The concretized values of `SlsAttributes` in such `RequirementsSets` are produced during the *requirements decomposition* activity (see the `decompose` method in the upcoming Subsection 5.5.3), which calculates the particular them based on the `RDRs` (see model no. 11 below) assigned to the particular `AbstractNodes`.

11. **`RequirementsDecompositionRules` (`RDR`)** model (utilized inside of the `NSST`):

An `RDR` is a grouping of `DecompositionRules` (see model no. 12) that describe how a single high-level SLS *Requirement* against a Subnet perceived as a whole shall be transformed into a set of its lower-level counterparts that can be posed against the `AbstractNodes` representing its internals. As such, `RDRs` convey the **domain knowledge that is essential for the automation of processing in the `decompose` method** (see method no. 3 in the upcoming Subsection 5.5.3). During the design process, the decomposed SLS

[153] NSSTs with a single `AbstractNode` representing a nested Subnet are utilized in the PoC for showcasing the translation of high-level SLS *Requirements* into their lower-level counterparts supported by the *Slicing Entities* managed in Subject Orchestrator – see Section 5.6.2 for details.

Requirements are used for evaluating the *candidate Slicing Entities* that are considered for mapping to a given `AbstractNode` (see the `evaluate` method in point 1 in the following Subsection 5.5.3).

In the PoC, `RDRs` of specifically crafted "translation NSSTs" are also utilized for **showcasing support for high-level SLS requirements in the input `SliceProfile`** in the case of lack of their explicit support in the utilized *Slicing Entities* and the Subject Orchestrator managing them (see Subsection 5.6.2 for details).

12. **`DecompositionRule`** model (utilized inside of an `RDR`)

A given `DecompositionRule` specifies the `translationMethod` from a higher-level *Requirement* to the lower-level one and the `valueCalculationMethod` for determining its value. Several examples of concretized `DecompositionRules` are provided in the upcoming Subsection 5.6.2.

13. **`InstanceDescriptor`** model:

`InstanceDescriptor` conveys concretized decisions about the structure of the designed Subnet. Instance of the `InstanceDescriptor` model is an in-memory object with a structure that is accepted by the phase (D) "New NSSI provisioning" procedures of the Subject Orchestrator. As such, `InstanceDescriptor` conveys the design decisions that were already enumerated in the context of the manual design process (see Subsection 5.4.1), specifying among others: reference of the utilized NSST, choice of concretized *Slicing Entities*, values of *Capabilities* that shall be reserved for utilization by the NSSI, values of *Capabilities* to be consumed by the input `SliceProfile`, choice of operational workflows, etc.

`InstanceDescriptor` is an ephemeral entity that exists only for a short moment between the end of the design process in phase C and the start of the deployment procedures in phase D of the allocation workflow. Thus, although `InstanceDescriptor` is a *Slicing Entity*, in the PoC it is not considered as a *candidate* in the subsequent design processes.

5.5.2.6. Models conveying results of the design activities

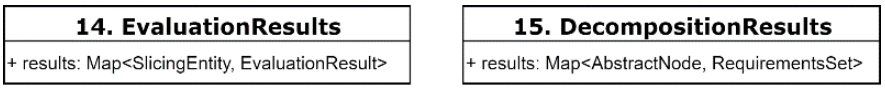


Figure 49. Models conveying results of selected design activities.

14. **`EvaluationResults`** model:

`EvaluationResults` convey information about the fulfillment of a `RequirementsSet` by the `CapabilitiesSets` of *Slicing Entities*, expressed as `scores` computed by the `evaluate` method (see method no. 1 in the upcoming Subsection 5.5.3). The `EvaluationResults` model contains a `score` for each *Slicing Entity* perceived as a whole, as well as `scores` for each of the evaluated *Requirement–Capability* pairs. A single `score` expresses not only a binary information about *Requirement* fulfillment (`FULFILLED`, `NOT_FULFILLED`), but also the degree of contribution towards its fulfillment, which is expressed by the following enum values:

- `NO_CAPABILITY` - when the evaluated *Slicing Entity* is not exposing a *Capability attribute* that would correspond to the posed *Requirement attribute*,
- `IRRELEVANT` – when a *Capability attribute* corresponding to the posed *Requirement attribute* exists, but its value is not present in the evaluated context (e.g. due to the fact that the *Capability* is not offered in the requested coverage area)
- `DOES_NOT_CONTRIBUTE` – when the *Capability attribute* exists, but its value is fully consumed (value is "zero" according to its data type) and such value does not contribute to the fulfillment of the posed *Requirement* value,
- `CONTRIBUTES` – when *Capability attribute* exists and exposes a "non-zero" value which does not meet the *Requirement* value fully, but is assessed as contributing to its fulfillment,
- `FULFILLS` – when the *Capability* offers a value which exactly matches the *Requirement* value,
- `EXCEEDS` – when the *Capability* offers a value which exceeds the *Requirement* value

The information conveyed in the `EvaluationResults` model (combined with the `DesignPolicies`, see model no. 16) is crucial for **enabling automated processing in the `select` method** described in upcoming Subsection 5.5.3 (see method no. 2).

15. **`DecompositionResults`** model:

Conveys the lower-level `RequirementsSets` resulting from the decomposition of a higher-level `RequirementsSet` and their assignment to the particular `AbstractNodes` in the utilized `NSST`. The information conveyed in the `DecompositionResults` model is crucial for **enabling automated processing in the `evaluate` method** described in upcoming Subsection 5.5.3 (see method no. 1).

5.5.2.7. Design process policies model

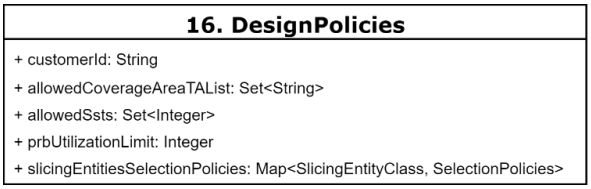


Figure 50. `DesignPolicies` model.

16. **`DesignPolicies`** model:

`DesignPolicies` are utilized to steer the logic of the ASD workflow. They are specified per Customer (PLMN), based on the results of business agreements with the Provider. `DesignPolicies` contain a set of settings utilized for validating the incoming NSSI allocation request as well as influencing the flow of processing in the end-to-end allocation workflow, aligning it with the operational objectives targeted by the Provider and its business relations with a particular Customer. `DesignPolicies` model allows for specifying:

- `allowedCoverageAreaTAList` – specifies a list of TAI[61] codes of geographical areas where NSSIs can be allocated by the Customer.
- `allowedSsts` - specifies a list of Slice types (SST[36]) supported by the NSSIs allocated by the Customer.
- `prbUtilizationLimit` - specifies the upper limit of PRB utilization in the NSSIs allocated by the Customer.
- `slicingEntitiesSelectionPolicies[SlicingEntity class]` – comprises a set of policies steering the selection of *Slicing Entities* in a given class, when considering evaluated *candidates* for mapping to a given `AbstractNode`:
  - `candidatesCount` – specifies if a `SINGLE` or `MULTIPLE` candidates of the given class should be selected. In the PoC, a `SINGLE` *Slicing Entity* is selected for a Subnet `AbstractNode` (NSSI or NSST) and `MULTIPLE` *Slicing Entities* can be selected for an MF `AbstractNode` (MFIs).
  - `evaluationThreshold` – specifies what evaluation score (see the `EvaluationResults` model no. 14) must be achieved by a *candidate* to be eligible for selection, e.g.: `FULFILLS`, `EXCEEDS` or `CONTRIBUTES`. In the PoC, `evaluationThreshold` for *candidates* for Subnet `AbstractNode` must be `FULFILLS` or higher (i.e. `EXCEEDS`). In turn, MFI *candidates* are selected with `evaluationThreshold` set to `CONTRIBUTES` or higher (i.e. `FULFILLS` or `EXCEEDS`).
  - `candidatesPriority` – specifies the order in which *Slicing Entities* in a given lifecycle stage are being considered:
    - `RUNTIME;DESIGN-TIME` – firstly attempt to select *candidate*(s) from the existing *Runtime Slicing Entities*. If such *candidate*(s) do not cross the `evaluationThreshold`, consider *Design-time Slicing Entities* for creation of a new Instance(s). For example, when considering *candidates* for a Subnet `AbstractNode`, firstly consider existing NSSIs. If the existing NSSIs do not fulfill the `SliceProfile`, attempt to design and deploy a new one by selecting an NSST.
    - `DESIGN-TIME;RUNTIME` – firstly attempt to select *candidate*(s) from the existing *Design-time Slicing Entities* (NSSTs) for designing a new Instance. If design is not successful, consider selecting from the existing *Runtime Slicing Entities*. For example, when considering *candidates* for a Subnet `AbstractNode`, firstly attempt to select an NSST for designing and deploying a new NSSI. If it is not possible to design a new NSSI meeting the `SliceProfile`, attempt to reuse an existing NSSI.
    - `RUNTIME` – select *candidate*(s) only from existing *Runtime Slicing Entities* that cross the `evaluationThreshold`.

  - ► `DESIGN-TIME` – select *candidate*(s) only from *Design-time Slicing Entities*.
- `runtimeCandidatesPriority` - specifies the priority criteria for selecting *candidate*(s) from multiple *Slicing Entities* that cross the `evaluationThreshold`:
  - ► `LOWEST_REQUIREMENTS_COUNT` – select a *candidate* with the highest count of assigned `RequirementsSets`.
  - ► `HIGHEST_REQUIREMENTS_COUNT` – select a *candidate* with the lowest count of assigned `RequirementsSets`.
  - ► `RANDOM` – select randomly from the fulfilling *candidates*.
  - ► `ALL` – select all eligible *candidates* (only compatible with `candidatesCount` = `MULTIPLE`).

The information conveyed in the `DesignPolicies` model (together with the `EvaluationResults`, see model no. 14) **is crucial for enabling automated processing in multiple activities of the end-to-end ASD design and allocation workflow,** including request validation as well as the operation of the `evaluate` and `select` methods described in the upcoming Subsections 5.5.3.1 and 5.5.3.2, respectively.

As seen from the specification above, the `DesignPolicies` model at its current stage comprises simple configurations that are statically specified by the Operator and Provider based on the agreements with the particular Customers. Extension of this model towards representing more complex rules possible to be evaluated by a policy engine is considered as a future work (see Subsection 5.7.3).

5.5.3. Individual design activities' implementation

The ASD PoC implements the design activities that are needed to fulfill the assumed PoC scope. The individual design activities are implemented inside of dedicated functional modules that expose high-level methods as depicted in Figure 51.

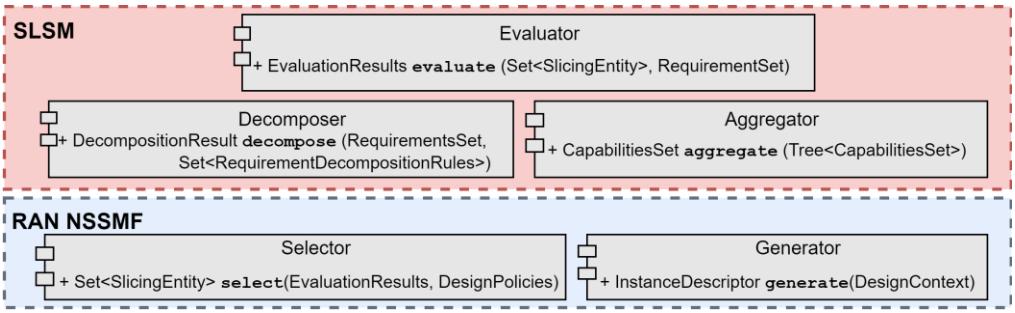


Figure 51. Functional modules containing implementation of selected design activities.

The **logic of processing in the implemented activities relies heavily on the features of the implemented models** which is strongly underlined in the previous Subsection. In particular, the **domain knowledge conveyed by these models is leveraged** as highlighted in the descriptions below:

5.5.3.1. Evaluation

`EvaluationResults` **`evaluate`**`(Set<SlicingEntity>, RequirementsSet)` method, provided by the "Evaluator" module of the SLSM microservice:

The `evaluate` method performs the (2) ***evaluation*** of the SLS *Requirements* fulfillment by the *candidate Slicing Entities*, by *comparing* and *matching* the `SlsAttributes` contained in the input `RequirementsSet` with the ones contained in the `CapabilitiesSets` exposed by the considered *Slicing Entities*. *Comparing* of a given pair of *Requirement-Capability* `SlsAttributes` and deciding if they *match* is steered by the `evaluationBehavior` defined in their corresponding `SlsDefinition` (see model no. 6 in Subsection 5.5.2.3). Based on the results of *comparing* and *matching*, `scores` of SLS *Requirements* fulfillment by the evaluated *Slicing Entities* are computed as the output `EvaluationResults`. This result conveys a single `score` produced for each *Slicing Entity* perceived as a whole, as well as detailed `scores` for each of their evaluated *Requirement-Capability* pairs.

5.5.3.2. Selection

`Set<SlicingEntity>` **`select`**`(EvaluationResults, DesignPolicies)` method, provided by the "Selector" module of the RAN NSSMF microservice:

Based on the `EvaluationResults` and `DesignPolicies`, the (3) ***selection*** of a *Slicing Entity* to be utilized in the further course of the design process is performed. When selecting *candidates* for a Subnet `AbstractNode`, a `SINGLE` *candidate* that fully satisfies the *Requirements* (with `score` either `FULFILLS` or `EXCEEDS`) is chosen, taking into account the other `slicingEntitiesSelectionPolicies` that have been set for a given Customer (see the `DesignPolicies` model no. 16 in Subsection 5.5.2.6). When selecting `MFIs`, `ALL` *candidates* evaluated with score `CONTRIBUTES` or higher are chosen to become the internals in the designed NSSI - see Subsection 5.6.5 for details.

5.5.3.3. Requirements decomposition

`DecompositionResults` **`decompose`**`(RequirementsSet, Set<RequirementDecompositionRules>)` method, provided by the "Decomposer" module of the SLSM microservice:

This method performs the (4) ***decomposition*** of a higher-level `RequirementsSet` into a set of its lower-lever counterparts. For each `SlsAttribute` of the input `RequirementsSet`, it executes the transformations according to the `translationMethod` and `valueCalculationMethod` specified in the input `RDRs` (see model no. 11 in Subsection 5.5.2.5). The resulting `DecompositionResults` contains the produced lower-level `RequirementsSets` to be posed against the `AbstractNodes` of the NSST utilized in the design process (see model no. 10 in Subsection 5.5.2.5).

Typically, the `decompose` method is utilized to transform the input `SliceProfile` into `RequirementsSets` to be posed against the *lowest-level Slicing Entities* that will build the designed NSSI internally. In the ASD PoC, the `decompose` method is also used to convert the high-level SLS *Requirements attributes* that are not natively supported by the *Slicing Entities* to the lower-level *Capabilities* that they support and expose (details provided in the upcoming Subsection 5.6.2).

5.5.3.4. Capabilities aggregation

`CapabilitiesSet` **`aggregate`**`(Tree<CapabilitiesSet>)`, provided by the "Aggregator" module of the SLSM microservice:

This method performs the (6) ***Capabilities aggregation*** activity[154], by merging multiple `CapabilitiesSets` of the lower-level *Slicing Entities* into a consolidated higher-level `CapabilitiesSet`. Aggregation is performed by executing the `capabilityAggregationBehavior` defined for the `SlsAttributes` contained in the input hierarchy of related `CapabilitiesSet` objects (represented as a tree), in a bottom-up manner. In the calculations, the `RequirementsSets` currently assigned to the `CapabilitiesSets` are taken into account, by prior execution of the `requirementAccumulationBehavior` and `capabilityConsumptionBehavior`, applied before the `capabilityAggregationBehavior`. Performing these calculations is graphically presented in Figure 46 in Subsection 5.5.2.4.

5.5.3.5. InstanceDescriptor generation

`InstanceDescriptor` **`generate`**`(DesignContext)`, provided by the "Generator" module of the RAN NSSMF microservice:

Based on the decisions and information gathered during the design process, the `generate` method prepares an in-memory `InstanceDescriptor` object representing design decisions with a structure that is accepted by phase (D) "New NSSI provisioning" procedures of the Subject Orchestrator (see model no. 13 in Subsection 5.5.2.5).

[154] Note: the aggregation of *Capabilities* of the designed NSSI is performed for its single, "primary" *Configuration Profile* only, which is created out of the single *Configuration Profiles* exposed by the MFI *Slicing Entities* being selected. Support for establishing the "alternative" *Configuration Profiles* is considered as an extension of the PoC in the future works (see point 2 in Subsection 5.7.3).

5.5.3.6. Determination of Capabilities share for supporting the input SliceProfile

In the PoC, there is no dedicated implementation of the (9) ***determination of Capabilities' share to allocate*** activity. Instead, the `decompose` method (described in Subsection 5.5.3.3) is reused for this purpose. More specifically, the values of the internal *Slicing Entities' Capabilities* to be allocated are taken exactly as the values of lower-level *Requirements* that are decomposed based on the `RDRs` contained in the NSST corresponding to the reused NSSI or the NSST utilized for designing the new NSSI. Implementation of a more advanced, dedicated method that could potentially decrease or increase the *Capabilities* share to allocate according to an enhanced `DesignPolicies` model is considered as a future work, described in Subsection 5.7.3.

5.5.3.7. Optimization of processing in selected design activities

It is worth noting that the PoC features the possibility of combining the `evaluate` and `select` methods with a feedback mechanism, which allows for speeding-up the processing of a potentially large set of *candidates* in the `evaluate` method. Once enabled, the selection criteria are checked after evaluating each *candidate* and the processing is interrupted as soon as a satisfying *candidate* is found.

Although optimizing the execution time of the design process (see the performance evaluation results in Subsection 6.4.5) such approach may be only utilized if it is thoughtfully decided that any of the satisfying *candidates* can be selected. If a more sophisticated selection mechanisms would be implemented (e.g. not only to determine "some" satisfying candidate but a one that satisfies the *Requirements* in the best way according to the criteria specified in the `DesignPolicies`) then the feedback mechanism would need to be disabled to allow the entire spectrum of *candidates* to undergo the evaluation.

Now, when the most important features of the implemented models and logic of the individual design activities have been explained, their utilization in the end-to-end NSSI allocation workflow can be presented.

5.6. Implementation of the ASD end-to-end NSSI design, provisioning and allocation workflow

In the previous Section, important implementation details of the individual models and design activities that are essential for enabling automated processing have been explained. In this Section, in turn, the composition of these elements into the ASD end-to-end automated NSSI design, provisioning and allocation workflow is discussed. Firstly, in Subsection 5.6.1, general organization of the ASD workflow is presented. Then, in Subsection 5.6.2, the prerequisites for its automated execution are discussed. Afterwards, a detailed walkthrough of the executed activities and models utilized in the particular phases of the workflow is provided:

- Subsection 5.6.2 explains how the support for some of the high-level SLS attributes in the input `SliceProfile` has been achieved thanks to the utilization of the automation framework concepts.
- In Subsection 5.6.4, the focus is on the part of the workflow that fulfills the assumptions of <AS1> by performing automated (2) *evaluation* and (3) *selection* of existing NSSI *candidates*.
- Subsection 5.6.5, in turn, explains how the workflow fulfills the assumptions of <AS2>, by automatically performing the design activities that result in the production of an `InstanceDescriptor` containing a *fully-concretized* specification for the provisioning of a new NSSI.

5.6.1. Organization of the ASD workflow

The implementation-level organization of the ASD end-to-end workflow is presented in Figure 52 and referred to extensively in the remaining content of this Section.

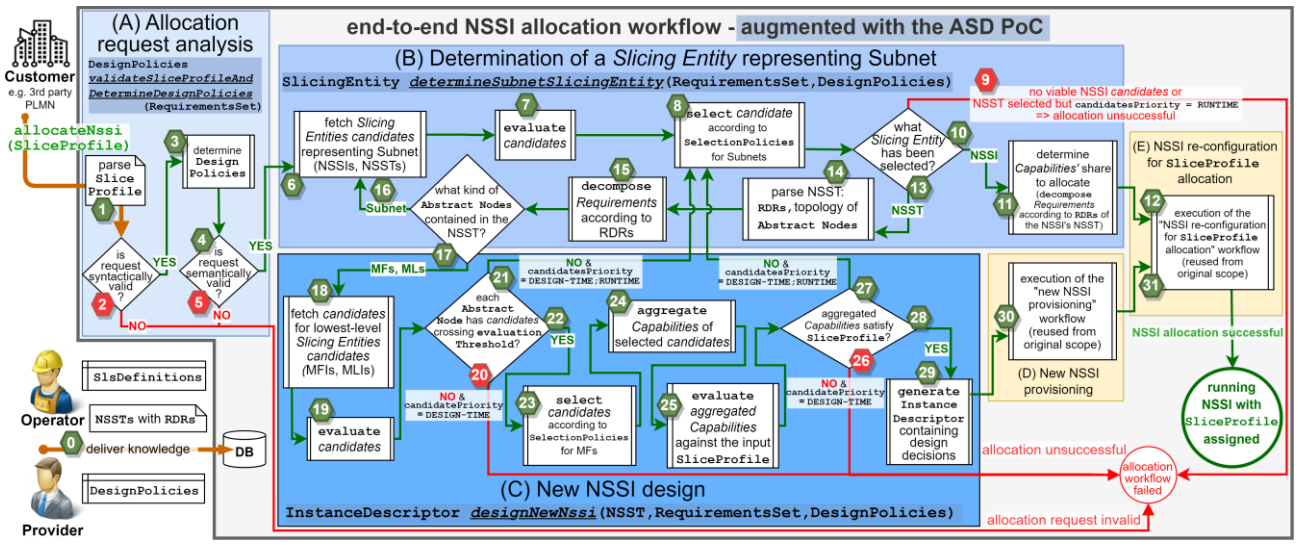


Figure 52. Organization of the end-to-end NSSI allocation workflow augmented with the ASD PoC.

As an operational prerequisite, several pieces of information need to be delivered to the system by the Operator and Provider in step 0, which is described in Subsection 5.6.2 that follows. The operation of the ASD workflow is triggered by an invocation of the `allocateNssi` operation by a Customer (e.g. a 3rd party PLMN) in Step 1. The execution of the individual design activities in phases (A), (B) and (C) are implemented in the main methods of the "ASD core" module (depicted in Figure 53 below), initiated in steps 1, 6 and 18, respectively:

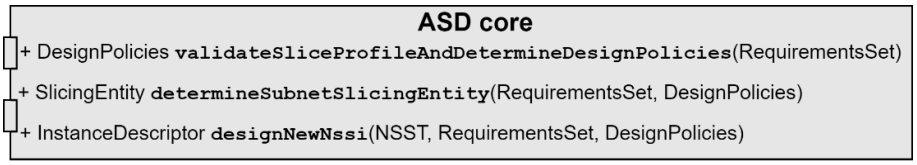


Figure 53. Methods implemented in the ASD core functional module, encompassing logic of the main flows of the ASD workflow.

Detailed organization of processing in these methods is described in the Subsections 5.6.3, 5.6.4 and 5.6.5, respectively, by referring to the flow depicted in Figure 52. In turn, processing in phases (D) and (E) reuses the procedures of the Subject Orchestrator that were already automated in its original state[155] - due to this fact their detailed discussion is omitted. It is worth mentioning that the entire ASD workflow is executed in a synchronous manner. It has been implemented this way in order to be strictly aligned with the specification of the `allocateNssi` operation in [31], which remains defined as a synchronous, blocking RESTful operation[17].

5.6.2. Prerequisites for automated execution of the ASD workflow

As an operational prerequisite (step 0), before accepting the allocation requests from the Customers, it is assumed that the required domain, operational and business knowledge has been captured from the Operator and Provider, and is deposited in the instances of models discussed in Subsection 5.5.2. In particular: `SlsDefinitions` for all supported `SlsAttributes` are specified and `NSSTs` equipped with appropriate `RDRs` are available in the files' repository. The concretized arrangement of instances of these models and the values conveyed by them need to be decided by the Operator based on the operational knowledge about the utilized *Slicing Entities* and the methods of processing of the SLS *Requirements*/*Capabilities* supported by the Orchestrator[156]. It is also assumed that the Operator and Provider established `DesignPolicies` based on business agreements with the Customers that are eligible for invoking the NSSI allocation requests (see model no. 16 in Subsection 5.5.2.6).

Once the aforementioned prerequisites are met, the Subject Orchestrator is **ready to handle the NSSI allocation requests in an automated manner, without the involvement of the human Operator**. It shall be noted that unlike in the original state of the Subject Orchestrator, the Operator does not need to prepare

[155] As described in Section 5.4.1, processing in phases (D) and (E) was already automated in the original state of the Subject Orchestrator.

[156] Several examples of configuring the aforementioned models with domain and operational knowledge have been provided in Subsection 5.5.2 and are also given in the upcoming Section 6.2.4, where the organization of the Case Study environment is discussed.

any pre-provisioned pool of NSSIs to handle the allocation requests. By making design decisions according to the specified `DesignPolicies`, the automated workflow will either design and deploy a new NSSI, or reuse one of the already existing ones, if available and configured to do so by the `DesignPolicies`. In particular, the automated workflow performs an on-demand creation of the necessary NSSIs even in a greenfield environment. Thus, pre-provisioning of NSSIs is no longer a prerequisite for supporting allocation requests, which was listed as one of the shortcomings of the Subject Orchestrator in its original state (see Subsection 5.4.2).

5.6.3. Automated SliceProfile validation and decomposition of high-level SLS Requirements (steps 1-8, 13-16)

The ASD workflow starts its operation in phase (A) "NSSI allocation request analysis" by invoking the **`validateSliceProfileAndDetermineDesignPolicies`**`(RequirementsSet)` method, which parses the input `SliceProfile` into the internal representation as a `RequirementsSet`, validating it syntactically (step 1). Failed validation interrupts the workflow (step 2). If successful, step 3, determines the `DesignPolicies` for the particular Customer based on the PLMN ID contained in the input `SliceProfile` (`pLMNInfoList.plmnId` attribute). Semantic validation of the request against `DesignPolicies` performed in step 4 verifies if the input `SliceProfile` contains `SlsAttributes` with values that are allowed to be posed by the invoking Customer. Among others, the requested Slice type specified in the `SliceProfile` (`pLMNInfoList.snssai.sst`[36] attribute) is checked to be among the `DesignPolicies.allowedSsts` for the requesting Customer and that the Slice coverage requested in the `SliceProfile.RANSliceSubnetProfile.coverageAreaTAList` is allowed for the Customer, which is specified in `DesignPolicies.allowedCoverageAreaTAList`. In case of any invalid `SliceProfile` content, the allocation procedure fails (step 5).

After successful validation, the workflow enters phase (B) "Determination of a *Slicing Entity* representing Subnet", by invoking the **`determineSubnetSlicingEntity`**`(RequirementsSet, DesignPolicies)` method of the "ASD core" module. In step 6, all the available *Slicing Entities candidates* representing Subnet are fetched: existing NSSIs from the RAN NSSMF inventory and NSSTs from the files repository. Then, in step 7, these *candidates* are evaluated against the SLS *Requirements* present in the input `SliceProfile`. At this moment it shall be reminded that the NSSI objects managed by the Subject Orchestrator expose *Capabilities* comprising only a limited set of high-level and a set of lower-level `SlsAttributes` (see Subsection 5.4.3 for details). Thus, if the input `SliceProfile` contains some high-level SLS *Requirements* that are not natively supported (e.g. `dLLatency`, `uLLatency`, `dLThptPerSlice`, `uLThptPerSlice`, `maxNumberofUEs`), the existing NSSI *candidates* are evaluated in step 7 as not fulfilling, due to lack of correspondence between the posed SLS *Requirements* and their offered *Capabilities* (`NO_CAPABILITY` evaluation result, see Subsection 5.5.2.6)

To overcome this limitation and augment the operational scope of the Subject Orchestrator, the support for the high-level SLS *Requirements* is achieved thanks to the multivariance of the conceptual framework, by establishing intermediate, so-called "translation NSSTs" that allow for decomposing the high-level `SlsAttributes` into their lower-level counterparts. Such NSSTs must be equipped with `RDRs` that contain the domain knowledge about the **correspondence between the high-level and low-level `SlsAttributes`**. In the real Subject Orchestrator's operation, such correspondence may depend on various conditions, e.g. features of the managed *Slicing Entities*, subsystems of the utilized Orchestrator, particularities of the deployment environment, etc.

In the PoC, it is assumed that the fulfillment of the high-level SLS *Requirements* regarding the guaranteed throughput per Slice (`dLThptPerSlice`, `uLThptPerSlice`), latency (`dLLatency`, `uLLatency`) and the count of supported UEs (`maxNumberofUEs`) is possible to be delivered based on the requested Slice type (`SliceProfile.pLMNInfoList.snssai.sst` attribute), specific fraction of PRBs assignment to the particular Slice (`prbPartitionMinRatioDl`, `prbPartitionMaxRatioDl`) and the priority of scheduling its traffic (`schedulingPriorityDl`, `schedulingPriorityUl`). To reflect this correspondence, the author crafted 3 "translation NSSTs" named "`EMBB_NSST`", "`MIOT_NSST`" and "`V2X_NSST`", each of them containing a single `AbstractNode` of type Subnet and a dedicated set of `RDRs` that express the aforementioned assumptions. Figure 54 below schematically depicts the structure of the "`EMBB_NSST`". Organization of the other "translation NSSTs" is similar.

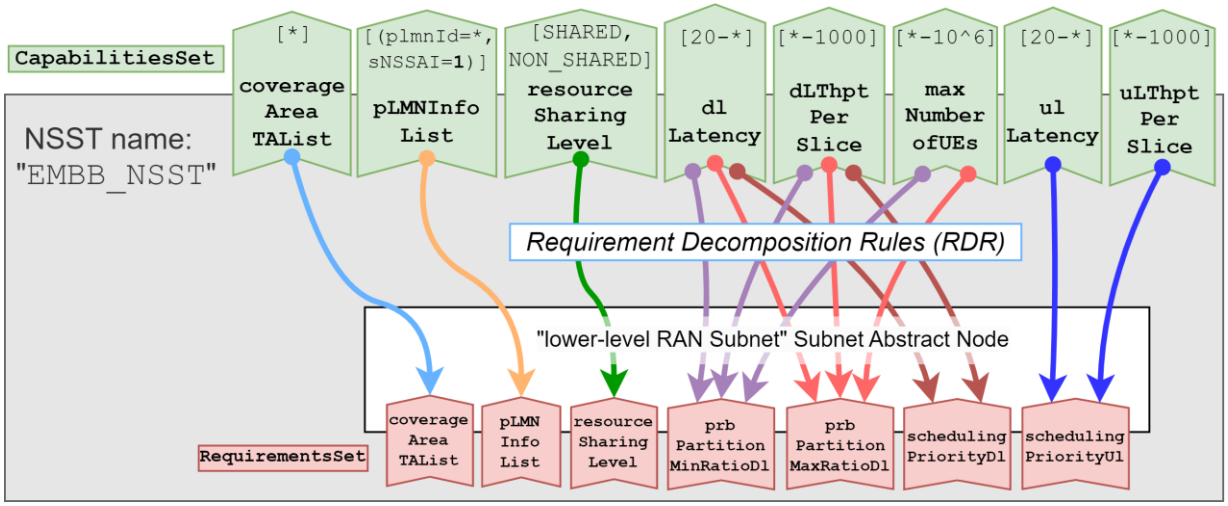


Figure 54. Organization of the EMBB_NSST "translation NSST" including the exposed *Capabilities* with high-level semantics and RDRs transforming high-level SLS *Requirements* into lower-level ones supported by the NSSI objects managed by the Subject Orchestrator.

The RDRs steering the decomposition of the high-level SlsAttributes in "EMBB_NSST", "MIOT_NSST" and "V2X_NSST" are presented in Tables 7, 8 and 9 below, respectively:

Table 7. *Requirements Decomposition Rules* (RDR) for high-level SLS *Requirements* against an eMBB-typed Subnet (SST=1).

high-level SLS *Requirements* in the input SliceProfile		RDRs to lower-level SLS *Requirements* supported by the *Slicing Entities* managed by the Subject Orchestrator			
SlsAttribute	value (v)	prbPartition MinRatioDl	prbPartition MaxRatioDl	scheduling PriorityDl	scheduling PriorityUl
dlLatency (ms)	20 <= v <= 30	20	**DesignPolicies. prbUtilizationLimit**	3	-
	30 < v <= 50	15		7	-
	50 < v <= 75	10		9	-
	75 < v <= 100	5		11	-
	100 < v <= 150	0		13	
	v > 150 or absence	0		-	-
ulLatency (ms)	20 <= v <= 30	-	-	-	3
	30 < v <= 50	-	-	-	7
	50 < v <= 75	-	-	-	9
	75 < v <= 100	-	-	-	11
	100 < v <= 150	-	-	-	13
	v > 150 or absence	-	-	-	-
dLThptPer SliceSubnet (Mbps)	600 < v <= 1000	15	**DesignPolicies. prbUtilizationLimit**	4	-
	300 < v <= 600	10		8	-
	100 < v <= 300	5		12	-
	v <= 100 or absence	0		15	-
uLThptPer SliceSubnet (Mbps)	600 < v <= 1000	-	-	-	4
	300 < v <= 600	-	-	-	8
	100 < v <= 300	-	-	-	12
	v <= 100	-	-	-	15
maxNumberofUEs	500 000 < v <= 1 000 000	15	**DesignPolicies. prbUtilizationLimit**	-	-
	300 000 < v <= 500 000	10		-	-
	50 000 < v <= 300 000	5		-	-
	v <= 50 000 or absence	0		-	-

Table 8. *Requirements Decomposition Rules* (RDR) for high-level SLS *Requirements* against an mIoT-typed Subnet (SST=3).

high-level SLS *Requirements* in the input SliceProfile		RDRs to lower-level SLS *Requirements* supported by the *Slicing Entities* managed by the Subject Orchestrator			
SlsAttribute	value (v)	prbPartition MinRatioDl	prbPartition MaxRatioDl	scheduling PriorityDl	scheduling PriorityUl
dlLatency (ms)	100 <= v <= 150	5	**DesignPolicies. prbUtilizationLimit * 50%**	6	-
	v > 150 or absence	0		-	-
ulLatency (ms)	100 <= v <= 150	-	-	-	6
	v > 150 or absence	-	-	-	-
dLThptPer SliceSubnet (Mbps)	100 < v <= 300	5	**DesignPolicies. prbUtilizationLimit * 50%**	7	-
	v <= 100	0		11	-
uLThptPer SliceSubnet (Mbps)	100 < v <= 300	-	-	-	7
	v <= 100	-	-	-	11
maxNumberofUEs	500 000 < v <= 1 000 000	15	**DesignPolicies. prbUtilizationLimit * 50%**	-	-
	300 000 < v <= 500 000	10		-	-
	50 000 < v <= 300 000	5		-	-
	v <= 50 000 or absence	0		-	-

Table 9. *Requirements Decomposition Rules* (RDR) for high-level SLS *Requirements* against an V2X-typed Subnet (SST=4).

high-level SLS *Requirements* in the input SliceProfile		RDRs to lower-level SLS *Requirements* supported by the *Slicing Entities* managed by the Subject Orchestrator			
SlsAttribute	value (v)	**prbPartition MinRatioDl**	**prbPartition MaxRatioDl**	**scheduling PriorityDl**	**scheduling PriorityUl**
dlLatency (ms)	5 <= v <= 15	20	**DesignPolicies. prbUtilizationLimit**	1	-
	15 < v <= 25	15		2	-
	25 < v <= 50	10		3	-
	50 < v <= 75	5		4	-
	75 < v <= 100	5		5	-
	v > 100 or absence	0		7	-
ulLatency (ms)	5 <= v <= 15	-	-	-	1
	15 < v <= 25	-	-	-	2
	25 < v <= 50	-	-	-	3
	50 < v <= 75	-	-	-	4
	75 < v <= 100	-	-	-	5
	v > 100 or absence	-	-	-	7
dLThptPer SliceSubnet (Mbps)	600 < v <= 1000	20	**DesignPolicies. prbUtilizationLimit**	3	-
	300 < v <= 600	10		7	-
	100 < v <= 300	5		11	-
	v <= 100 or absence	0		14	-
uLThptPer SliceSubnet (Mbps)	600 < v <= 1000	-	-	-	3
	300 < v <= 600	-	-	-	7
	100 < v <= 300	-	-	-	11
	v <= 100 or absence	-	-	-	14
maxNumberofUEs	50 000 < v <= 100 000	15	**DesignPolicies. prbUtilizationLimit**	-	-
	20 000 < v <= 50 000	10		-	-
	10 000 < v <= 20 000	5		-	-
	v <= 10 000 or absence	0		-	-

It is worth noting that according to the RDRs specified in Tables 7, 8 and 9, a single high-level SlsAttribute (e.g. maxNumberofUEs) may be decomposed into several lower-level SlsAttributes (e.g. prbPartitionMinRatioDl and prbPartitionMinRatioDl). As such, depending on the particular set of high-level SlsAttributes posed by the Customer in the input SliceProfile, the result of the initial *Requirements decomposition* may produce a set of overlapping lower-level SlsAttributes. Thus, as a final part of the decomposition process, the values of such overlapping lower-level SlsAttributes are aggregated according to the requirementAccumulationBehavior specified in their SlsDefinitions and placed into the resulting RequirementsSet.

Table 10. Examples of lower-level RequirementsSets produced in the initial *Requirements decomposition* activity.

SlsAttributes in SliceProfile with SST=1 (eMBB)	DecompositionResults
ulLatency = 25ms	**schedulingPriorityUl = 3**
dlLatency = 30ms	**prbPartitionMinRatioDl = 20** **prbPartitionMinRatioDl = DesignPolicies.prbUtilizationLimit** **schedulingPriorityDl = 3**
dlLatency = 40ms **dlThptPerSlice = 500Mbps** **ulThptPerSlice = 200Mbps** **maxNumberofUEs = 250000**	**prbPartitionMinRatioDl = 15** **prbPartitionMinRatioDl = DesignPolicies.prbUtilizationLimit** **schedulingPriorityDl = 7** **schedulingPriorityUl = 12**

Table 10 presents several examples of the input SliceProfiles obtained from the Customer and the DecompositionResults that will be utilized in the further course of the design process. By utilizing such an approach, if high-level SLS *Requirements* are posed in the SliceProfile, then thanks to a wide range of offered *Capabilities* values, one of the aforementioned "translation NSSTs" that matches the requested SST will be evaluated in step 7 as fulfilling the SliceProfile, and, consequently, selected in step 8. In such case, the workflow proceeds with step 13 to parse the NSST in step 14. In each "translation NSST", there is a single AbstractNode towards which the decompose method executed in step 15 produces lower-level RequirementsSet based on "translation NSST" RDRs (see Tables 7, 8 or 9). Finally, due to the fact that the single AbstractNode is of type Subnet, the workflow proceeds with step 16 towards recursive invocation of the **determineSubnetSlicingEntity**(RequirementsSet, DesignPolicies) method. The subsequent processing in this method (initiated again in step 6) is based on the decomposed lower-level SLS *Requirements* and is described in Subsection 5.6.4 that follows.

At this point, a disclaimer regarding the presented *Requirements decomposition* mechanism must be clearly stated, which relates to the caveats discussed in Subsection 5.1.3. Due to the fact that the high-level SLS *Requirements* are represented by their lower-level counterparts, the NSSI designed in the further course of the process (steps 6-31 described in Subsection 5.6.4 that follows) is actually composed to meet the lower-

level SLS *Requirements*, as only such *attributes* are explicitly supported by the utilized *Slicing Entities*. As such, the Subject Orchestrator's runtime SLS assurance mechanisms and the *Slicing Entities* themselves stay unaware of the fact that the high-level SLS *Requirements* have been posed by the Customer. Thus, in runtime, only their lower-level reflection is checked for fulfillment without guaranteeing that this will entail fulfillment of the high-level expectations.

For the reasons stated above, the fulfillment of the high-level SLS *Requirements* in NSSI runtime may only be credible if the correspondence between the high-level *Requirements* and the low-level *Capabilities* has been verified by extensive experimental tests of the utilized *Slicing Entities* with the particular configurations, in various operational loads. Again, it shall be reminded that the fulfillment of the high-level SLS *Requirements* in the RAN domain can be most credibly achieved by continuous, adaptive manipulation of the *Slicing Entities*' low-level configurations performed by dedicated near-RT and non-RT RIC components (see explanation in Section 5.4.3). The SLS assurance, however, is a broad research area that addresses the NSI/NSSI operation phase which is out of scope of this thesis.

Bearing the abovementioned facts in mind as well as the PoC caveats described in Subsection 5.1.3, the *Requirements decomposition* based on high-level NSSTs with `RDRs` configured as described above has been implemented in the PoC purely for the demonstration purposes, in order to highlight the multivariance of the elaborated framework and to showcase how its elements may be flexibly composed to establish an operational workflow that extends the Subject Orchestrator's operational scope in a particular context. When implementing the automation framework inside of another Orchestrator, a different variant of design activities arrangement can be applied, adjusting the flow of processing to the operational goals to be achieved.

Once the abovementioned steps are accomplished and the lower-level SLS *Requirements* are determined, the workflow continues its operation with 16 towards recursive execution of step 6, which is described in the following Subsection.

5.6.4. Automated evaluation and selection of an existing NSSI (steps 6-12)

In this Subsection, the operation of the ASD workflow in method **`determineSubnetSlicingEntity`** `(RequirementsSet, DesignPolicies)` is described, but executed with lower-level SLS *Requirements*. Firstly, all the *Slicing Entities* representing Subnet are again fetched (step 6) and evaluated for the fulfillment of the `RequirementsSet` (step 7). However, this time, thanks to the correspondence of the lower-level SLS *Requirements* to the *Capabilities* offered by the *candidates*, all the existing, running NSSIs as well as the NSSTs that exhibit lower-level *Capabilities* have the chance of being evaluated as satisfying them. The "translation NSSTs" will be evaluated as not fulfilling, since the `SlsAttributes` offered in their `CapabilitiesSet` do not correspond to the `SlsAttributes` of lower-level semantics that are now contained in the input `RequirementsSet`.

The `EvaluationResults` calculated in the `evaluate` method are analyzed in step 8 for making decision about the further course of the workflow - the `select` method chooses from among the NSSIs and NSSTs *candidates* that cross the `evaluationThreshold`, by following `DesignPolicies.slicingEntitiesSelectionPolicies` specified for *Slicing Entities* of class Subnet. If the `candidatesPriority` is set to `RUNTIME` only and there are no NSSI *candidates* evaluated as `FULFILLS`, the workflow is unsuccessful (step 9). If the `candidatesPriority` is set to `RUNTIME;DESIGN-TIME`, and there is an NSST that fulfills the low-level SLS *Requirements*, then it is attempted to design a new NSSI based on such NSST - this scenario is discussed in Subsection 5.6.5 that follows.

When the `candidatesPriority` is set to `RUNTIME;DESIGN-TIME` (default) or only `RUNTIME`, and there is at least one existing NSSI *candidate* evaluated as crossing the `evaluationThreshold`, then a `SINGLE`[157] *candidate* is selected according to the `runtimeCandidatesPriority` policy (step 10). This policy may be variously configured, depending on the Operator's operational objectives and the results of business negotiations of

[157] In the current operational scope, SliceProfile may be assigned only to a `SINGLE` NSSI, which is currently the only possible value of the candidatesCount policy in case of *Slicing Entities* representing Subnet. Support for selecting more `MUTIPLE` ones, serving as "backup", may be considered in Subject Orchestrators with more advanced implementations.

the Provider with the Customer. By setting it to `LOWEST_REQUIREMENTS_COUNT`, the ASD workflow is configured to balance the NSSIs' load incurred by the assigned `SliceProfiles`. Alternatively, to maximize resources utilization, the `HIGHEST_REQUIREMENTS_COUNT` policy may be set, which will result in the preferred reuse of NSSI *candidates* that are the most loaded, but still offer *Capabilities* with values that are sufficient to accommodate the incoming `RequirementsSet`[158]. As a last step 11, the share of internal *Slicing Entities' Capabilities* to be devoted for allocating the input `SliceProfile` is determined by executing the `decompose` method with the `RDRs` of the NSST corresponding to the selected NSSI. The produced `DecompositionResults` are taken as the values of *Capabilities* that will be consumed in the internal *Slicing Entities* of the selected NSSI.

Afterwards, based on the produced design outcomes (pointer to the selected NSSI and the specification of *Capabilities* to be allocated), the activities of the (E) "NSSI re-configuration for `SliceProfile` allocation" phase are executed in the selected NSSI (step 12). Here, the logic present in the original operational scope is reused (see Figure 39 in Subsection 5.4.2). After the execution of phase (E) is accomplished, the entire end-to-end allocation workflow is successful.

5.6.5. Automated design of a new NSSI (steps 7-31)

If among the *candidates* evaluated in step 7 there are no NSSIs fulfilling the SLS *Requirements*, or if the `slicingEntitiesSelectionPolicies[Subnet].candidatesPriority` is set to `DESIGN-TIME` only, the `select` method in step 8 may determine an NSST which exposes lower-level *Capabilities* that fulfill the input `RequirementsSet`. In such case, the workflow follows step 13 and parses the content of the selected NSST in step 14. The obtained topology of internal `AbstractNodes` and `RDRs` feed the `decompose` method executed in step 15, which transforms the `RequirementsSet` posed against the entire Subnet into distinct `RequirementsSets` against the *candidates* that will be evaluated for substituting the internal `AbstractNodes`. Since the internal `AbstractNodes` are not representing a Subnet anymore, but the *lowest-level Slicing Entities* (MFs and MLs) out of which a new NSSI should be designed, the workflow follows step 17 entering phase (C) "New NSSI design", executing the `designNewNssi(NSST, RequirementsSet, DesignPolicies)` method of the "ASD core" module.

Logic executed in this method attempts to produce an `InstanceDescriptor` of an NSSI that will satisfy the input `RequirementsSet` based on the structure of `AbstractNodes` dictated by the input `NSST` and by following the `DesignPolicies`. Firstly, in step 18, the `MFI` *candidates* and `MLIs` connecting them are fetched from the NFMF inventory. The `MFIs` are passed together with the decomposed `RequirementsSets` to the `evaluate` method (step 19), where the decomposed `RequirementsSets` are compared and matched against the `CapabilitiesSets` of the `MFIs` *candidates*, by taking into account the `evaluationBehavior` of the contained `SlsAttributes`. In the returned `EvaluationResults`, all `MFI` *candidates* are assigned with `scores` indicating the degree of *Requirements* fulfillment. Besides `MFIs` evaluation with the generic `evaluate` method, the topology of `MLIs` between the `MFIs` is checked in a dedicated manner[159] to verify that the evaluated *candidates* match the `NSST`'s topological assumptions of the NSSI to be designed.

After evaluation, the `EvaluationResults` are analyzed to verify that at least one *candidate* possible to be mapped to each `AbstractNode` exists. If any of the `AbstractNodes` is left without a suitable *candidate* and the `slicingEntitiesSelectionPolicies[Subnet].candidatesPriority` is set to `DESIGN-TIME` only, the design of a new NSSI is unsuccessful (step 20). But, if the `candidatesPriority` is set to `DESIGN-TIME;RUNTIME`, the workflow proceeds back to step 8, where the already existing NSSI *candidates* will have the chance of being selected and utilized as explained in the previous Subsection 5.6.4.

If there are *candidates* crossing the `evaluationThreshold` for each `AbstractNode` (step 22), the `select` method is invoked in step 23 to choose the `MFI` *candidates* together with their interconnecting `MLIs` based on the

[158] The influence of various DesignPolicies on the course of the ASD workflow and, as a consequence, on the produced design results in various operational circumstances is showcased in Section 6.3, as a part of functional PoC evaluation.

[159] As explained in Subsection 5.5.2.2, MLIs *Capabilities* are not supported in the Subject Orchestrator. The topology indicated by the MLI objects are utilized for verifying the relationships established between the MFIs.

`EvaluationResults` and `slicingEntitiesSelectionPolicies[ManagedFunction]`. In the PoC, `MULTIPLE` (`candidatesCount`) `RUNTIME` (`candidatesPriority`) MFI *candidates* can be selected for each `AbstractNode`.

After selection, the `aggregate` method is invoked (step 24) to calculate the aggregated `CapabilitiesSet` of the designed NSSI's internal composition, which is performed according to the `capabilityAggregationBehavior` of each `SlsAttribute`[160]. Afterwards, the *aggregated* `CapabilitiesSet` of the designed `NSSI` is evaluated for input `RequirementsSet` fulfillment by a final call to the `evaluate` method (step 25), followed by analyzing the returned `EvaluationResults`. If the *aggregated* `CapabilitiesSet` does not fulfill the input `RequirementsSet` and the `slicingEntitiesSelectionPolicies[Subnet].candidatesPriority` is set to `DESIGN-TIME` only, the design of a new NSSI is unsuccessful (step 26). Otherwise, if the `candidatesPriority` is set to `DESIGN-TIME;RUNTIME`, the workflow proceeds with step 27 which, similarly to step 21, will attempt selection from among the already existing NSSI *candidates*. `EvaluationResults` indicating input `RequirementsSet` fulfillment (step 28) trigger the generation of an `InstanceDescriptor` (step 29) which ends the execution of the `designNewNssi` method.

Next, the `InstanceDescriptor` returned from the `designNewNssi` method is utilized as input to the procedures of the (D) "New NSSI provisioning" (step 30) and the `DecompositionResults` produced in step 15 are utilized as values of internal *Slicing Entities' Capabilities* to be allocated in the (E) "NSSI re-configuration for `SliceProfile` allocation" (step 31) phases. These already automated provisioning and allocation procedures are reused from the original scope of the Subject Orchestrator (see Subsection 5.4.1). Once the NSSI and its internals are fully provisioned and configured, the end-to-end NSSI allocation workflow is successfully finished.

5.7. PoC implementation summary

This Chapter presented a PoC implementation of a Network Slice Subnet design automation solution that complies with the concepts of the conceptual framework elaborated in Chapters 3 and 4. The Subject Orchestrator and its augmentation with the ASD PoC have been detailed, including the description of the implemented models and the working mechanisms of the end-to-end, fully automated NSSI design, provisioning and allocation workflow. In the Subsections below, the key takeaways from the ASD PoC implementation are presented. Firstly, the achieved functional augmentation is summarized by referring to the shortcomings of the Subject Orchestrator that have been successfully resolved in the PoC. Then, several important conclusions regarding the technical realization of the automation framework are stated. The provided observations are supportive in the verification of the thesis statements, which is discussed in the upcoming Section 7.1, after presenting results from the empirical experimentations in Chapter 6.

5.7.1. Achieved functional augmentation

The implementation of the PoC allowed for a significant functional augmentation of the Subject Orchestrator and elimination of its shortcomings enumerated in Section 5.4, as justified in the following paragraphs.

The part of the ASD workflow executed in steps 6-12 (described in Subsection 5.6.4) corresponds to the operational scope originally supported by the Subject Orchestrator, namely the allocation of the already existing NSSIs. However, the difference is that thanks to the design activities automation, the `NSSI` candidate does not need to be pointed by the Customer when invoking the `allocateNssi(SliceProfile, NssiId)` operation - the currently available NSSI candidates are now **automatically evaluated and a suitable one is automatically selected** by the ASD mechanisms. As a result, the signature of the `allocateNssi(SliceProfile)` operation is now fully standards-compliant. Moreover, **the Customer does not need to possess any knowledge** about the NSSIs currently deployed in the Provider's infrastructure and does not need to make any selection decisions, making utilization of the `allocateNssi` operation purely declarative. This part of the ASD PoC fulfills the assumptions of <AS1>.

In turn, automation of the design activities in steps 7-31 (described in Subsection 5.6.5) fulfill the assumptions of <AS2>, by enabling **dynamic design and deployment of new NSSIs** at the moment of obtaining the incoming `allocateNssi` requests. As a result, the Operator **does not need to design new NSSIs manually** and

[160] MLIs do not participate in the aggregation, as they do not expose *Capabilities* in the Subject Orchestrator.

the NSSIs **do not need to be pre-provisioned** before the invocations of the `allocateNssi` operation by the Customers. Thanks to such features, the Subject Orchestrator now bears the hallmarks of offering "dynamic slicing" operation. But most notably, these enhancements allow for **eliminating the need of human Operator participation** in the design process and, consequently, in the entire allocation pipeline.

Lastly, the spectrum of **high-level SLS *Requirements*** supported in the input `SliceProfile` has been **extended**. This further enhances the declarative nature of Subject Orchestrator's operation from the Customer's perspective. It is because instead of specifying *Requirements* in terms of low-level, radio medium-related aspects, the Customer can now express its demands utilizing tangible, high-level measures that are oriented on the Service that will run on the Slice.

Thus, it can be summarized that from the functional perspective the implemented solution fully meets the PoC implementation scope defined in Subsection 5.1.2. The verification of its expected runtime functioning, fulfilling the functional requirements stated in Section 2.4 is provided in the upcoming Section 6.3, where the functional evaluation of the elaborated concepts is discussed.

5.7.2. Conclusions related to the technical realization of the conceptual framework

Several conclusions related to the technical realization of the proposed conceptual framework, relevant in the context of thesis verification, can be formulated already at this stage of the dissertation, still before discussing the experimental evaluation of the PoC in Chapter 6.

Detailed description of the implementation aspects provided in the current Chapter gave an in-depth and broad insight into how the theoretical concepts of the proposed framework can be implemented in practice, **proving their feasibility**. As shown in this Chapter, implementation of the framework's concepts allowed for establishing models and methods that constitute well-defined, **generic units** with **clear responsibilities** and **complementing roles**. The feasibility of composing an end-to-end automated design workflow out of such individual elements is a direct, desired consequence of the framework's **internal coherency**. The conceptual framework is also **coherent externally**, since all of its elements are fully aligned to the Network Slicing provisioning standards in force.

The framework's elements can be **assembled in various variants**, depending on the operational scope and architectural assumptions of the Orchestrator being augmented. The conceptual framework defines several backward-compatible Automation Scopes (<AS>) that can serve as baselines for determining a suitable subset of elements to be implemented in the particular circumstances. As an example, in the ASD PoC the Subject Orchestrator has been augmented with the elements assumed by <AS1>, and gradually enhanced with the elements specified in <AS2>. An important conclusion drawn from the PoC is that the extent of framework's concepts possible to be implemented in a particular Orchestrator **depends inextricably on the current state of its development, architectural arrangements and the advancement of the offered functionalities**[161]. Hence, implementing <AS3> in the Subject Orchestrator was not possible, as it misses the integration between the NFV-MANO and Slicing orchestration blocks, which is a prerequisite for implementing <AS3> (see Subsection 5.3.2).

Furthermore, the concepts building up the proposed framework allowed for constructing a system that is **multivariant.** In the PoC, a particular variant of the ASD workflow's execution can be achieved by configuring the values of attributes in the generic models. As the most illustrative example, configuration of the `DesignPolicies` allows steering the flow of allocation procedures to support the Provider's and Operator's objectives in the particular business and operational circumstances. It is worth noting that such runtime-level multivariance enables adjusting the operation of the augmented Orchestrator to the needs of various Providers/Operators by configuring the variant of the workflow's execution **without the need of changing the solution's source code**. As another example, the support for the high-level SLS *Requirements* in the input `SliceProfile` is achieved by leveraging the generic *Requirements decomposition* mechanism that is fed with

[161] As it will be shown in the upcoming Section 6.4.5, this dependency is also valid in terms of the design process performance, as it is largely influenced by the efficiency of the mechanisms already implemented in the Subject Orchestrator.

"translation NSSTs" containing a customized variant of `RDRs` which, instead of targeting lower-level `AbstractNodes` (MFs, MLs), result in the production of lower-level *Requirements* that can be posed against NSSIs managed in the Subject Orchestrator.

Another feature of the conceptual framework is that it allows for constructing system exhibiting high **structural scalability**, which in [169] is defined as *the ability of a system to expand in a chosen dimension without major modifications to its architecture*. Indeed, by leveraging the modeling principles of abstraction and separation of concerns, the framework's elements are **generic** and **extendable** which greatly contributes to the ASD workflow's structural scalability. These virtues are clearly visible in the case of models utilized for SLS management and the hierarchy of *Slicing Entities* models, as justified in the paragraphs below.

The generic `SlsAttribute` model and its accompanying `SlsDefinition` enable uniform and extendable modeling of *Capabilities* and SLS *Requirements* that exhibit heterogenous semantics, simply by configuring their behaviors. Such a modeling and architectural approach **prioritizes configuration over implementation**, contributing to increased reuse of the generic models and limiting the need of coding the customized ones. As a result, providing support for new `SlsAttributes` can be achieved by configuring the generic models, without the need for any implementation-level changes (if the already supported behaviors can be reused). Actually, support for the high-level SLS *Requirements* has been added in the PoC simply by delivering configuration of their behaviors to the instances of the `SlsDefinition` model and by supplying the "translation NSSTs" with appropriately specified `RDRs`, which required no changes in the Subject Orchestrator's source code.

As another example, the proposed hierarchy of abstract *Slicing Entities* models' separates the operation of the design activities from the highly-specific details of heterogeneous *candidates* considered to be utilized in the designed Slice/Subnet – enforcement of `CapabilitiesSet` exposition by each *Slicing Entity* assures their **uniform processing in the design activities**. The spectrum of considered *candidates* can be extended by inheriting from one of the generic, abstract *Slicing Entities* models. In such way, support for NSSTs in the automated design process has been added in the PoC, which did not require large implementation efforts thanks to the reuse of the already existing models. Similarly, adding support for other *Slicing Entities* (e.g. MF or ML Descriptors, considered as a future work) can be achieved.

5.7.3. PoC implementation enhancements considered as a future work

Despite the achieved functional extensions, it must be noted that the ASD PoC at its current stage of development cannot be considered a "reference implementation" of the conceptual framework elaborated in this thesis, as it does not implement its full extent. The ASD PoC also does not solve all the design-time related operational deficiencies of the Subject Orchestrator. Thus, there is room for enhancements that may result in further extensions of the supported Automation Scopes and the use cases covered by the PoC. Moreover, further development of the PoC may serve as a basis for investigation of research aspects that are not the main subject of this dissertation, but may constitute its continuation.

In the list below, several aspects that are considered for improvement in the scope of future works are enumerated. However, it shall be noted that the extent of framework's concepts possible to be implemented in a particular Orchestrator being augmented depends inextricably on the current state of its development and the advancement of the offered functionalities, as concluded in the previous Subsection. Hence, progressing in several of the enumerated directions requires prior improvements to be applied in the Subject Orchestrator.

1. Due to lack of integration between the NFV-MANO and the Network Slicing components in the Subject Orchestrator, the *lowest-level Design-time Slicing Entities*, like MF or ML Descriptors, are not supported in the ASD workflow. While this is in-line with the Subject Orchestrator's original operational purpose (see Section 5.3.2), the <AS3> or <AS4> cannot be realized without prior integration of the aforementioned blocks. Such enhancement requires significant, long-lasting effort of a team of developers, which was beyond the capabilities of the author alone while working on this dissertation. For this reason, <AS3> and, possibly, <AS4> are considered for implementation as a future work, depending on the further development directions of the Subject Orchestrator.

2. In the PoC, the concept of *Configuration Profiles* is not explicitly modeled, because the simplifying assumption proposed in Subsection 4.7.1 has been applied. Consequently, the *Runtime Slicing Entities* (MFIs, NSSIs) being evaluated and selected during the design process are considered "as is", without taking into account any other *Configuration Profiles* that could be potentially exposed by them. Consequently, the configurations in the generated NSSI *Instance Descriptor* can be equated to the single "primary" *Configuration Profile*. The PoC could be extended to include the consideration of other *Configuration Profiles* of MFIs as *candidates* in the evaluation and selection activities. Selection of a different MFI *Configuration Profile* than currently exposed would entail MFI's reconfiguration in phase (E). It would also allow the generation of "alternative" *Configuration Profiles* in the produced *Instance Descriptor* and generation of SLS assurance policies to switch between them, paving the way towards realization of <AS3>.
3. Currently, exposition of *Capabilities* by the MLI objects and endpoints (`EP_RPs`) is not supported in the Subject Orchestrator. This prevents considering their contribution to the fulfillment of the input SLS *Requirements* during the design process. Once such support is implemented, the PoC may be extended to perform the evaluation and selection of such *Slicing Entities* when deciding on the composition of a newly designed Subnet, especially if it would be designed from *Design-time Slicing Entities* (achievable in <AS3>).
4. The `DesignPolicies` implemented in the ASD PoC are simple, and their choice is limited. For example, the currently implemented `slicingEntitiesSelectionPolicies.runtimeCandidatesPriority` takes into account only the count of `RequirementsSets` assigned to a *Slicing Entity* (e.g. `HIGHEST_REQUIREMENTS_COUNT`). But a more indicative strategy of selection should take into account the accumulated (aggregated) load induced by the assigned `RequirementsSets`. It is because there might be multiple `RequirementsSets` assigned that "consume" as much *Capabilities* as a single `RequirementsSet` that is very demanding. Another enhancement is to enable steering the share of *Capabilities* to be devoted for supporting the `SliceProfile` by specifying their increase/decrease policies, including support for overbooking. Additionally, more advanced logic may be implemented to make the policy choice dependent on some conditions, e.g. the status of the Customer, global state of the infrastructure, etc. Such more complex policies could be processed by a dedicated policy engine added as another component to the Subject Orchestrator's architecture.
5. Currently, the ASD PoC is supporting automated design of NSSIs at the level of a single administered RAN domain. Collective administration of multiple RAN domains may be performed by the Provider by means of a "uber-NSSMF" which could serve as an umbrella on top of RAN NSSMFs orchestrating the particular domains. Such "uber-NSSMF" could include the implementation of a variant of the ASD workflow that would design NSSIs with a global outlook of the infrastructure resources available in the covered RAN domains. By performing such a centralized design process in the "uber-NSSMF", a single higher-level NSSI could be allocated, which internally would consist of the lower-level NSSIs allocated in the particular domains. In such case, deployment environments'' *Capabilities* models would need to be implemented and the (2) *evaluation* activity would need to be enhanced to support their processing.

Chapter 6

Experimental evaluation

The description of the ASD PoC implementation described in Chapter 5 already showcases several features of the elaborated conceptual framework presented in this thesis, including its feasibility, coherency, genericity, extendibility, flexibility and multivariance. However, to fully verify the embodiment of the framework concepts against the thesis statements and the topic of this dissertation, a thorough empirical evaluation of the constructed PoC solution needs to be conducted. This Chapter describes the experimentations that allow for such a positioning.

Firstly, in Section 6.1, the main goals of PoC evaluation are stated. Next, organization of the experiments is discussed in Section 6.2, including their scope, methodology, testbed and assumptions. Then, in Section 6.3, evaluation of the PoC solution from the functional perspective is presented. Finally, performance evaluation of the PoC is described in Section 6.4.

6.1. Goals of PoC experimental evaluation

The primary goal of PoC experimental evaluation is to **obtain measurable indicators supporting verification of the thesis statements**. Since the thesis is concentrated on the functional research aspect, the experimentations must focus on the **functional evaluation** of the PoC solution, in order to verify that the implemented system **indeed enabled automating the design process**, allowing for hands-free, end-to-end handling of the incoming allocation requests without Operator' involvement. Showcasing that this is achieved in a spectrum of conditions resembling a potential real Network Slicing deployment will serve as an experimental verification of the thesis statements.

Moreover, to strengthen the verification of the functional aspect and to gain a more comprehensive insight into the operational characteristics of the developed solution, it should also be evaluated **in the increasingly demanding load conditions**, including its assessment from the **non-functional perspective**. The goal is not only to verify that the implemented PoC solution functions properly with the growing scale of operated NSSIs, but also to assess **how well it performs**. Based on the collected indicators, the operational **maturity of the developed solution may be assessed**, revealing its bottlenecks, operational limits and identifying areas for improvement. Finally, based on the collected results, **judgement of industrial applicability** of the elaborated concepts should be attempted as well.

6.2. Organization of the experiments

To meet the goals defined in the previous Section, the empirical evaluation of the PoC has been divided into two parts: (1) **functional verification** described in Section 6.3, followed by (2) **performance evaluation** described in Section 6.4. Before proceeding to the description of the conducted experiments, several important assumptions and caveats related to their organization are clarified in the following Subsections, including the scope of conducted experiments, the nature of the utilized input data and the configuration of the experimental testbed.

6.2.1. Scope

Functional verification of any computer program may be performed in various extents. Today's dominant practice in software engineering is to verify program's correct functioning empirically. The ASD PoC elements have been tested by the author for correct functioning at various levels, including code-level unit and integration testing executed in a "white box" manner, as well as end-to-end system testing performed in a

"black box" manner. However, it is both impracticable and undesirable to present the details of all such tests in this thesis.

Instead, to facilitate intuitive presentation, the basic functional verification of the elaborated concepts is showcased based on a series of end-to-end, "semi black-box" functional tests organized into a **Case Study** described in Section 6.3. The Case Study exercises the PoC solution for the delivery of expected results in a representative set of scenarios resembling a potential real Network Slicing deployment in a sample RAN domain. Such organization of the Case Study allows for demonstrating desirable functioning of the automated ASD workflow on its main execution paths, showcasing its most important features from the perspective of thesis verification.

In addition to the Case Study, a reinforced functional verification combined with **performance evaluation** is presented in Section 6.4. In these experiments, the PoC solution is further examined for proper functioning with different sets of input data of growing complexity, and a spectrum of performance measurements is collected. Analysis of the obtained results allows for revealing how efficiently the ASD workflow as a whole and its particular parts handle the allocation requests, when the scale of NSSI-to-be-allocated and the scale of NSSIs already managed by the Subject Orchestrator change.

It is also important to remind that the scope of the thesis and the conducted experiments is solely on the design-time aspects, **not on the deployment nor runtime issues**. The design process responsibility is solely to produce decisions that will appropriately configure the control-plane of the *Slicing Entities*, preparing them for the operation with new set of *Requirements*. However, how the allocated NSSI and its particular internals perform in runtime, and in particular **whether they deliver the expected level of SLS satisfaction for the user-plane traffic**, is **beyond the scope of the conducted experiments**, which has been specified in the thesis scope in Section 1.6. For the aforementioned reasons, although the entire end-to-end allocation workflow is executed in each of the conducted tests, the main area of interest is the processing in the design activities executed in phases (A), (B), (C) and the produced design results that configure the control-plane of the operated gNB installations.

6.2.2. Methodology

The methodology of experiments conducted in both parts of the evaluation revolves around invoking the `allocateNssi(SliceProfile)` operation, followed by the observation of the ASD PoC mechanisms' operation and the effects it brings in the state of the orchestrated *Slicing Entities*. However, the attention of these observations is on different aspects in the Case Study and in the performance evaluation, as clarified in the following paragraphs.

The Case Study is focused on a **step-by-step analysis of the executed design activities**, **verifying their expected functioning in the particular circumstances**. What is verified are the intermediary and final results produced by the ASD mechanisms, namely the results of evaluation, selection, decomposition and aggregation activities and, most importantly, the finally selected one of the existing NSSIs or the structure of the newly produced Subnet `InstanceDescriptor`. The correctness of these results is verified by checking their alignment to the configured `DesignPolicies` and the **fact that they are accepted as correct inputs** by the "New NSSI provisioning" and "NSSI re-configuration for `SliceProfile` allocation" procedures executed by the Subject Orchestrator in phases (D) and (E). The fact that these procedures execute successfully based on the delivered input, allowing for hands-free execution of the entire end-to-end allocation workflow, serves as the ultimate verification of the ASD PoC proper functioning[162].

In turn, the second part of the evaluation is focused on the **collection of a spectrum of performance indicators and their analysis**. The collected results are studied for discovering the computational complexity of the individual design activities, of the design process as a whole, as well as of the entire end-to-end allocation pipeline. In particular, the analysis focuses on **identifying the factors** that influence the performance of these processes. This allows for assessing if the timeframe of ASD workflow's operation in

[162] Depending on the experienced operational conditions and the `DesignPolicies` configured for handling the particular allocation requests, there may be various results produced by the design process that can be perceived correct. This aspect is further discussed in Section 6.3.5.

the increasingly complex load conditions can be perceived sensible from the Customer's as well as Operator's and Provider's perspectives.

What also differentiates these two parts of evaluation is the construction of the **input data** utilized for the experiments[163]. For the Case Study, the inputs have been arranged to demonstrate ASD PoC operation in a scenery that mimics a potential Network Slicing deployment in a sample, small scale RAN administrative domain. In turn, for the performance evaluation, the input data is organized to reflect a larger scale RAN domain in which various arrangements of preexisting NSSIs and `SliceProfiles` assigned to them are assumed. This allows for evaluating the ASD mechanisms in **various load conditions** of the Subject Orchestrator. Detailed description of the inputs utilized in the basic and extended tests is provided in the corresponding Subsections: 6.3.2 and 6.4.2, respectively.

6.2.3. Assumptions and caveats

In the context of the input data utilized in the experiments, the nature of the data representing the arrangement of gNB installations and resources in the administered RAN domain requires commenting. At the moment of this thesis writing, practical realization of the Network Slicing technology in commercial deployments remain a carefully guarded trade secret. Unfortunately, the author was unable to get access to operational data from any of the ongoing Network Slicing trials. For these reasons, the experiments are conducted in **synthetic** environmental conditions that are simulated by means of an **EMS Simulator** which allows for reflecting the anticipated organization of sliceable RAN resources (gNB installations) to the best of the author's knowledge. More on EMS Simulator and its role in the conducted experiments is provided in the following Subsection 6.2.4, which describes the experimental testbed.

It also needs to be clearly stated that the spectrum of inputs feeding the Subject Orchestrator, their possible arrangements and values, is huge. Moreover, the conditions of operational inception of the elaborated solution would vary between Subject Orchestrator deployments in different domains, with different equipment and with different operational objectives of a particular Provider. Testing the PoC solution for the entire spectrum of possible input values and the combination of all the possible states of environmental conditions is infeasible to be performed even by a large development team. Nevertheless, the author believes that the spectrum of operational conditions utilized in the experiments is sufficient for validating the legitimacy of the thesis statements.

Another organizational remark is that although the conducted tests could be concentrated solely on the design activities executed in phases A-C, by executing them in isolation from the phases that follow (see discussion of experiments scope in Subsection 6.2.1), the author decided that **each test case executes the entire allocation workflow**, i.e. all phases A-E. Such organization of the tests enables not only validating correct functioning of the ASD mechanisms, which is the main merit of the thesis, but also analyzing the changes incurred by the design decisions in the operated infrastructure and the effects they induce in the state of operational objects managed by the Subject Orchestrator.

Lastly, it is worth noting that neither the Case Study nor the performance tests aim at examining how the solution handles multiple allocation requests at the same time. Simultaneous invocation of this operation is not supported by the Subject Orchestrator – such requests are queued and processed sequentially one at a time. Thus, in the experimentations, **single execution** of the `allocateNssi` operation at a time is considered. Several other remarks related specifically to the organization of the performance evaluation are provided in Subsection 6.4.1.

6.2.4. Testbed

The organization of the ASD PoC deployment in the experimental testbed is schematically depicted in Figure 55 and described in the paragraphs below.

[163] There is a large range of inputs feeding different components of the Subject Orchestrator and the systems it interacts with, e.g. the ME objects in EMS that represent the orchestrated gNB installations, the state and configuration of the corresponding *Runtime Slicing Entities*, preexisting NSSIs, SlsAttributes present in the input `SliceProfile`, `DesignPolicies` utilized for handling the allocation requests, etc. See Subsections 6.3.2 and 6.4.2 for detailed description of the inputs utilized in the particular experiments.

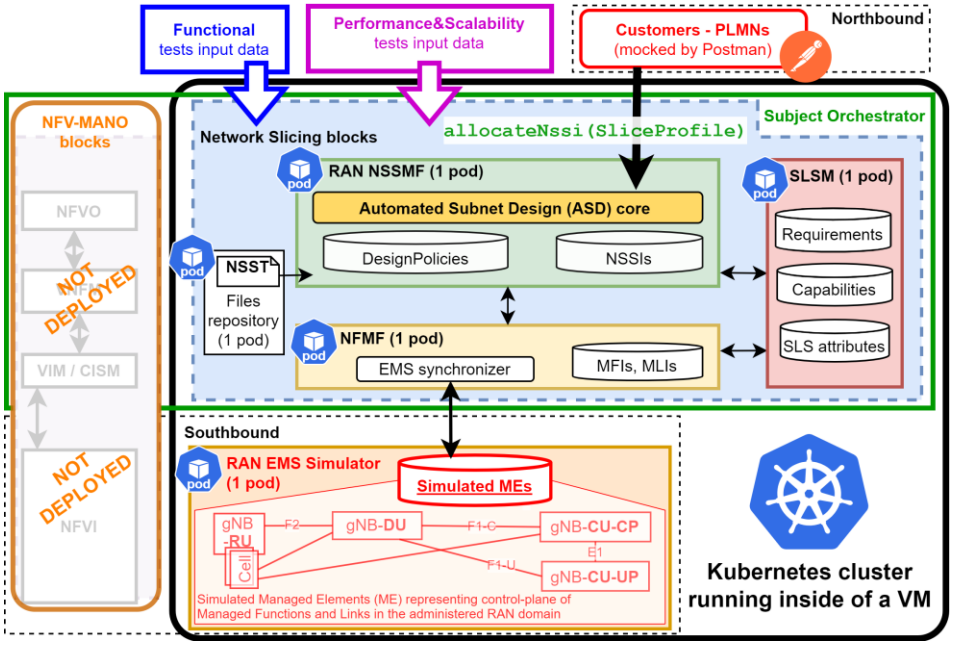


Figure 55. Organization of the experimental testbed and input data feeding the Subject Orchestrator.

Due to the synthetic conditions in which the experiments are conducted (refer to the assumptions described in the previous Subsection), in the southbound the Subject Orchestrator interacts with an **EMS Simulator** instead of real EMS and gNB installations. The role of the EMS Simulator is to mimic the operation of the ME objects, which represent the control plane of gNB installations in the administered RAN domain. EMS Simulator allows for representing any arrangement of the gNB MEs in the managed RAN domain by feeding it with different sets of input data. As such, it can be flexibly configured for various scenarios assumed in the particular experiments.

EMS Simulator exposes an API that is identical to the real EMS. This allows the Subject Orchestrator to synchronize the MEs information[164] and transmit the Slicing-related configuration to the MEs, according to the design decisions produced by the ASD mechanisms. Utilization of the EMS Simulator eliminates the need for interacting with real gNB MEs that in an operational environment would run as applications inside of VNFs deployed in the NFVI environments (OpenStack or Kubernetes clusters). Thus, the NFVI components could be excluded from the experimental testbed (see lower left part of Figure 55). Moreover, since spawning new VNFs is not supported within <AS2>, the NFV-MANO orchestration blocks (NFVO, VNFM, VIM/CISM) have been excluded from the testbed deployment as well.

For conducting the two parts of evaluation, the Subject Orchestrator's inventories and the EMS Simulator is fed with different sets of input data, which is schematically depicted with the blue and purple rectangles in the top of Figure 55. For the Case Study, the input mimics greenfield conditions of a sample RAN domain. In turn, the performance tests assume a RAN domain with various arrangements of brownfield scenarios, in which some NSSIs and `SliceProfiles` assigned to them already exist. Thus, the conditions utilized in the execution of the particular test cases have been firstly generated in a semi-automated manner and serialized into database dump files. Then, such dumps have been used for feeding the operational repositories of the Subject Orchestrator in order to guarantee repeatable execution of experiments in identical conditions. Detailed descriptions of the input data utilized for the functional and performance evaluation are provided in Subsections 6.3.2 and 6.4.2, respectively.

At this moment it should be clarified, that the utilization of an EMS Simulator does not lower down the credibility of the performed experimentations. It is because the design activities that are the main subject of the experiments are essentially high-level procedures that do not directly interact with the low-level ME objects, nor with the EMS. Instead, the design activities operate on high-level *Slicing Entities* objects only, which encapsulate all the lower-level aspects of their operation. For this reason, utilization of the EMS Simulator fed with mocked MEs operational data is sufficient for a credible evaluation of the ASD automated

[164] Automated EMS->NFMF synchronization involves creation and tracking of the MFI and MLI *Slicing Entities* in NFMF and their corresponding *Capabilities* in SLSM - see Subsections 5.3.1 and 5.4.1 for details.

design mechanisms that are executed in phases (A), (B) and (C) of the end-to-end allocation workflow. It should be noted, though, that the use of the EMS Simulator does influence the times of execution in phases (D) and (E). It is because instead of interacting with real gNB installations in the target deployment environment, these procedures are executed on the in-memory mocked ME objects. Thus, the measured times of their execution will certainly differ from real deployments. However, procedures executed in phases (D) and (E) do not concern the design activities and, as such, are out of the main interest in the experimentations.

For appropriate isolation of the experiments' execution, the Subject Orchestrator's components have been deployed inside of a dedicated Kubernetes cluster. Each component runs in a single pod[165] and all interactions between the tested components are performed inside of the same, single cluster. Only the NSSI allocation requests triggered from Postman[166] are incoming from the outside of the cluster, crossing the system's NorthBound interface boundary through the Kubernetes cluster's Ingress. The Kubernetes cluster in version 1.21.10 is embedded inside of a Virtual Machine hosted on a powerful physical machine equipped with 2x Intel Xeon Gold 5118 CPUs with 2,3GHz clock and 5MB cache, 256 GB of RAM and fast SSD hard disks organized into RAID 0 matrix. The machine is running a CentOS Linux 7 installation utilizing KVM virtualization for hosting VMs. The subject VM hosting the dedicated Kubernetes cluster is statically assigned with 6 vCPUs and 20 GB of vRAM. Besides the subject VM, 3 VMs of similar configuration are deployed on the physical machine. However, these VMs do not overlap the underlying physical resources.

6.3. Part 1 - functional evaluation – a Case Study

In the following Subsections, the most important aspects of the Case Study are described. Firstly, the considered operational scenery is presented. Next, the organization of the input data that determines the operational conditions in which the experiments are executed is described. Then, specification of Scenarios and Test Cases considered in the Case Study is presented, followed by the description of the most important details of their execution. Finally, conclusions from the Case Study are stated to summarize the obtained results.

6.3.1. Operational scenery

The Case Study considers Network Slicing orchestration performed by an independent Provider who owns a 5G Standalone physical and virtualized RAN infrastructure in a given wide-area territory. In the covered locations, the Provider is offering standards-aligned, declarative allocation of RAN NSSIs with selected QoS guarantees for 3rd party PLMNs (e.g. MVNOs) who construct end-to-end Slices (NSIs) to be offered to their clients - the NSSIs allocated in the Provider's premises become elementary building blocks of such NSIs.

The focus of the Case Study is on a single RAN domain administered by the Provider, located on the outskirts of a large city, as schematically depicted in Figure 56.

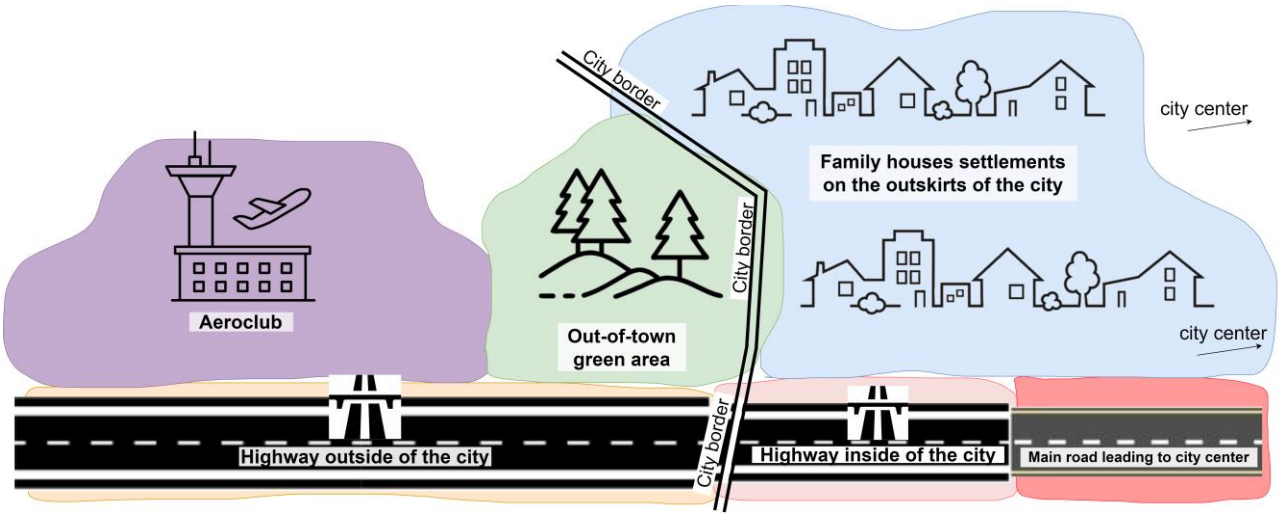


Figure 56. Geographical overview of the administered RAN domain located on the outskirts of a large city.

[165] The ultimate aim of the experimental evaluation is to test the PoC solution itself, not the runtime platform in which it is deployed (Kubernetes). This perspective is further clarified in Subsection 6.4.1.

[166] Postman (https://www.postman.com) is an HTTP client that is commonly utilized for testing RESTful APIs.

For the management of NSSIs in the considered RAN domain, a single installation of the Subject Orchestrator augmented with the ASD PoC is utilized. The territories outside of the covered area are managed by other RAN administrative domains that are out of scope of consideration in the Case Study.

6.3.2. Operational conditions of the experiments

For the organization of the Case Study, the Subject Orchestrator is fed with input data that determines its initial state, mimicking the operational conditions in the managed RAN domain. The input data schematically depicted in Figure 57 with letters A, B and C is described in Subsections 6.3.2.1, 6.3.2.2 and 6.3.2.3, respectively.

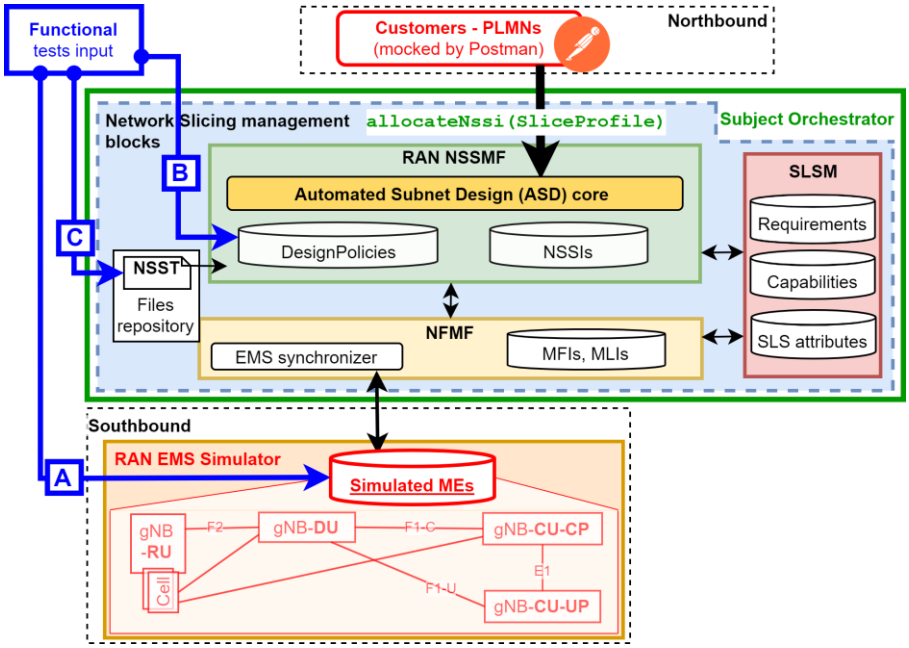


Figure 57. Inputs feeding the experimental testbed utilized for the functional evaluation of the PoC solution in a Case Study scenario.

Afterwards, in Subsection 6.3.2.4, the initial state of *Runtime Slicing Entities* objects managed in the Subject Orchestrator's operational inventories (NSSIs, MFIs, *Capabilities*, *Requirements*) is explained. Clear indication of the *Runtime Slicing Entities*' initial *Capabilities* state allows for tracking changes incurred in their values by the subsequently handled NSSI allocation requests, which will help verifying correct operation of the ASD PoC mechanisms at the particular stages of the Case Study execution.

6.3.2.1. Control-plane of the RAN infrastructure – gNB MEs synthetic data

As a result of RAN site planning, Operator divided the administered domain territory into 7 Tracking Areas[61] that comprise areas inside and outside of the city, as schematically depicted in Figure 58 and described below:

- TA1 covers the territory in the proximity of an aeroclub located outside of the city, where the civil aircrafts and Unmanned Aerial Vehicles (UAVs) are expected to appear.
- TA2 covers part of highway up to the border of the city, where the outer-city car traffic is expected.
- TA3 covers highway inside of city premises, where the increasingly dense car traffic is expected.
- TA4 covers the ending of the high-way and its transition to the main road leading to the city center.
- TA5 covers the green recreational area outside the city.
- TA6-TA7 are covering different parts of family houses settlements located in the city suburbs.

The synthetic environmental data that feeds the EMS Simulator (input A in Figure 57) mimics the control-plane configuration of 4 logical gNB installations[167] that provide radio coverage in the enumerated areas as shown in Figure 56. These gNB installations are identified with letters A-D and comprise the following MEs:

[167] Due to the CUPS separation of the gNB MFs in 5G (see Subsection 2.1.2), the count of logical gNBs is not an indicative measure of a RAN domain's scale. A single gNB might encompass multiple internal MFs that cover a vast area, e.g. one CU-CP can control 100s of DUs, constituting a single logical gNB.

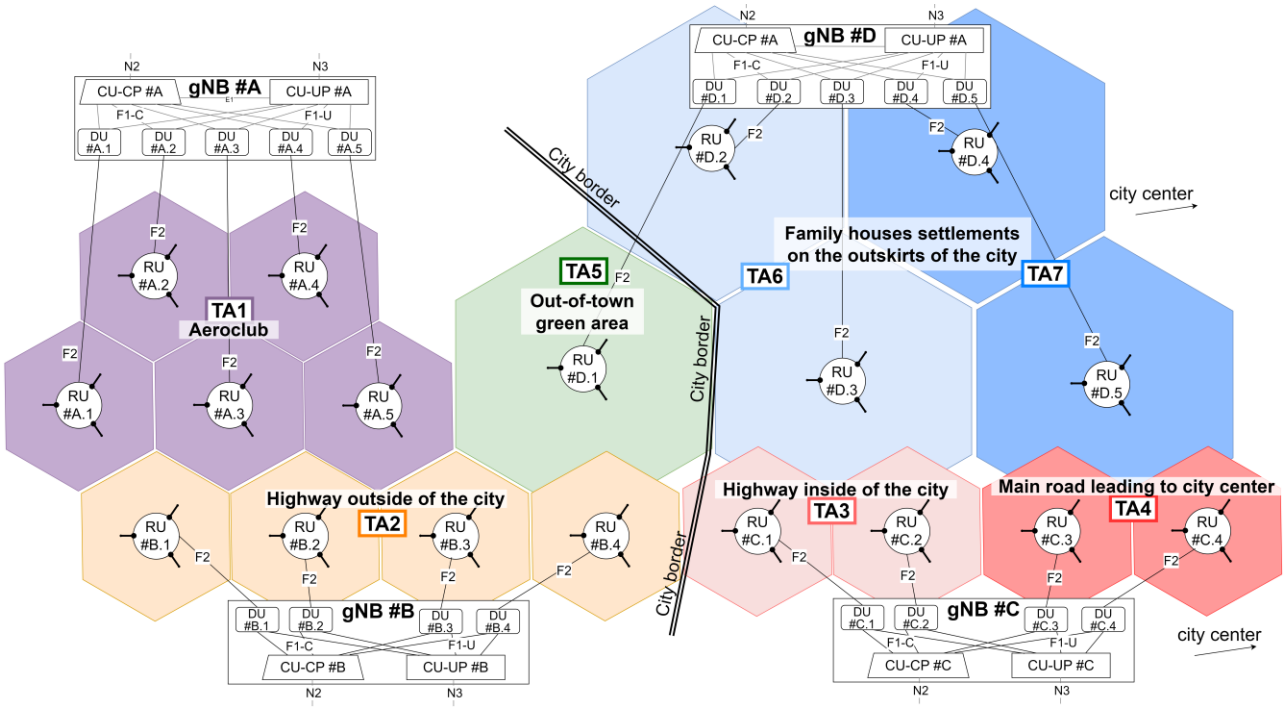


Figure 58. Organization of gNB installations in the administered RAN domain - location of RUs and topology of Links between gNB MEs.

- gNB#A is composed of: CU-CP#A, CU-UP#A and DUs #A.1 – 5 connected to RUs #A.1 – 5 that provide coverage in Cells belonging to TA1.
- gNB#B is composed of: CU-CP#B, CU-UP#B and DUs #B.1 – 4 connected to RUs #B.1 – 4 that provide coverage in Cells belonging to TA2.
- gNB#C is composed of: CU-CP#C, CU-UP#C and DUs #C.1 – 4 connected to RUs #C.1 – 4 that provide coverage in Cells belonging to TA3 and TA4.
- gNB#D is composed of: CU-CP#D, CU-UP#D and DUs #D.1 – 5 connected to RUs #D.1 - 5 that provide coverage in Cells belonging to TA5, TA6, and TA7.

Every RU comprises 3 Remote-Radio-Heads (RRH) organized into a common three-sector configuration, with each RRH serving a 120-degree sector. As such, in the considered scenario there are in total 26 gNB MEs: 4 CU-CPs, 4 CU-UPs and 18 DUs[168] that support 18 Cells. For simplicity, it is assumed that each Cell is handled by a single RU/DU – in other words, there is no simultaneous coverage of the same Cell by two distinct DUs/RUs inside of the same RAN domain. Within each gNB installation, appropriate connectivity by means of MLIs has been established: F2-U and F2-C in the fronthaul, F1-C, F1-U and E1 in the midhaul as well as N2 and N3 backhaul connectivity towards the MEs in the Core domain (AMFs and SMFs, respectively).

Coverage of the family houses settlements and the out-of-town green area is handled by gNB#D, which operates micro-cells in the mid-frequency bands (2,5GHz) that provide coverage of ~1-2 km radius on average in the suburban conditions. Cells managed by gNBs #A-#C, which are located along the highway and in the neighborhood of the aeroclub, are operated as pico-cells with higher frequency bands (3,5 GHz). Higher frequencies allow them to offer lower latencies and higher data-rates, but limit their range of coverage to ~500m in radius in the open-air conditions. Such configuration of the RAN resources differentiates the gNB installations by the types of Slices (SST) possible to be allocated in the covered areas. More specifically, *Slicing Entities* comprising gNB#A-C, due to the offered low latencies, support the allocation of eMBB, mIoT and V2X-typed Slices (SST values 1, 3 and 4, respectively), whereas gNBs #D supports eMBB and mIoT Slice types (SST values 1, 3).

An important operational assumption established by the Operator is that **all gNB installations in the described RAN domain are dedicated exclusively for Slicing**. As such, on the same territory, there might be other gNBs or pre-5G NB/eNB installations that support the traditional, general purpose "one-size-fits-all" network (which might be managed by another Provider, potentially utilizing a non-slicing orchestration stack).

[168] RUs are physical MEs not explicitly represented in the MFIs repository – they are represented by their corresponding DU *Slicing Entities*.

Consequently, it is assumed that **full spectrum of PRBs of the managed Cells is available** for the allocation of NSSIs. For brevity and clarity reasons, description of other low-level configuration of the particular Cells, gNB MFIs and MLIs (i.e. exact operational frequencies, channels, configuration of protocols, etc.) is omitted.

Based on all the decisions regarding RAN site planning described above, the Operator deployed the gNB installations and supplied their bootstrap configuration utilizing the NFV-MANO and EMS systems[169] - the synthetic data about MEs that feeds the EMS Simulator reflects all the described arrangements. The *Capabilities* of MFI *Slicing Entities* created by the Subject Orchestrator based on these MEs are discussed in the upcoming Subsection 6.3.2.4.

6.3.2.2. DesignPolicies

In the Case Study, it is assumed that all *Slicing Entities* in the administered domain support the MORAN feature[170] - this allows for their simultaneous utilization by multiple PLMNs, enabling more cost-efficient exploitation of the RAN infrastructure. By taking advantage of MORAN, the Provider established business partnerships with two PLMNs: X and Y[171]. The business negotiations resulted in the establishment of `DesignPolicies` (input B in Figure 55), presented in Table 11.

Table 11. `DesignPolicies` utilized in the Case Study.

DesignPolicies	**PLMN X**	**PLMN Y**
`allowedCoverageAreaTAList`	`[TA1-7]`	`[TA2-7]`
`allowedSsts`	`[1, 3, 4]` (eMBB, mIoT, V2X)	`[1, 3]` (eMBB, mIoT)
`prbUtilizationLimit`	`80%`	`60%`
`- slicingEntitiesSelectionPolicies [class = **Subnet**]:`		
`candidatesCount`	`SINGLE`	
`evaluationThreshold`	`FULFILLS`	
`candidatesPriority`	`DESIGN-TIME;RUNTIME`	`RUNTIME;DESIGN-TIME`
`runtimeCandidatesPriority`	`LOWEST_REQUIREMENTS_COUNT`	`HIGHEST_REQUIREMENTS_COUNT`
`- slicingEntitiesSelectionPolicies [class = **Managed Function**]:`		
`candidatesCount`	`MULTIPLE`	
`evaluationThreshold`	`CONTRIBUTES`	
`candidatesPriority`	`RUNTIME`	
`runtimeCandidatesPriority`	`ALL`	

As a premium Customer, PLMN X is treated preferentially. The value of the `allowedCoverageAreaTAList` policy indicates that PLMN X can allocate NSSIs in the entire area covered by the domain, whereas PLMN Y is not allowed to allocate NSSIs in the aeroclub neighborhood (TA1). The value of the `allowedSsts` policy indicates that PLMN X is allowed to allocate NSSIs for all types of Slices offered by the Provider in the covered locations: eMBB, mIoT and V2X (SST values 1, 3 and 4). In turn, PLMN Y does not have the possibility of utilizing the high-cost gNB capabilities for offering V2X - only allocation of NSSIs with types eMBB and mIoT (SST values 1 and 3) is allowed for PLMN Y. Lastly, utilization of PRBs as a crucial RAN resource is regulated by the `prbUtilizationLimit` policy. According to the business agreements, PLMN X is allowed for utilizing 80% PRBs at peak demands (only if not currently consumed by other PLMNs). PLMN Y, in turn, is set with 60% limit for peak demands.

Policies for selection of *Slicing Entities* have been established for both PLMNs, expressed in the `slicingEntitiesSelectionPolicies`. In the case of Subnets, the `candidatesCount` and `evaluationThreshold` are common for both PLMNs: a `SINGLE` *candidate* that `FULFILLS`[172] the posed *Requirements* should be selected. Policies that differ among the two PLMNs determine the priorities of Subnet *Slicing Entities* selection. In the case of PLMN Y, `candidatesPriority` set to `RUNTIME;DESIGN-TIME` favors reusing the already existing NSSIs, rather than creating new ones. Moreover, `runtimeCandidatesPriority` set to `HIGHEST_REQUIREMENTS_COUNT` indicates that among the NSSI candidates fulfilling *Requirements*, the one which has the highest count of assigned `SliceProfiles` should be selected – such setting allows for maximizing the utilization of existing NSSIs.

[169] The deployment and bootstrap configuration of gNB installations is performed in a manual/assisted manner, as assumed by the Subject Orchestrator's operational scope described in Subsections 5.3.2 and 5.4.1.

[170] See Subsection 5.3.3 for the explanation of MORAN in terms of gNB MFs' *Capabilities* attributes.

[171] For the sake of simplicity, simple PLMN names are utilized: X and Y instead of full `PLMNInfo` structures containing MCC/MNC codes.

[172] If overbooking of *Slicing Entities* would be supported than lower levels of *Requirements* fulfillment could be considered. However, overbooking is out of scope in the PoC, but considered as its future extension, as described in Subsection 5.7.3.

In contrast, with the `candidatesPriority` set to `DESIGN-TIME;RUNTIME`, creation of a new NSSI is preferred for PLMN X. An existing NSSI candidate will be considered for reuse only if there is no possibility of creating new NSSI. Moreover, among the existing NSSI candidates considered for reuse, the ones with the least count of assigned `SliceProfiles` (`LOWEST_REQUIREMENTS_COUNT`) will be selected. As such, PLMN X is treated preferentially for allocating NSSIs with dedicated resources. Policies for selecting Managed Functions *Slicing Entities* are the same for both PLMNs: `ALL RUNTIME` candidates (MFIs) that `CONTRIBUTE` to *Requirements* fulfillment should be selected.

6.3.2.3. NSSTs

For the experiments, Subject Orchestrator is equipped with the set of 4 NSSTs (input C in Figure 55): three "translation NSSTs" named "`EMBB_NSST`", "`MIOT_NSST`" and "`V2X_NSST`" as well as the "`LOW_LEVEL_RAN_NSST`". The structure of the "translation NSSTs" is as explained in Subsection 5.6.2. The structure of "`LOW_LEVEL_RAN_NSST`" is as explained in Subsection 5.5.2.5.

6.3.2.4. State of operational objects

The input data utilized in the Case Study represents greenfield operational conditions - the gNB installations are operational, but no NSSIs exist yet and no `allocateNssi` requests have arrived yet from the PLMNs, thus no `SliceProfiles` are registered in the system yet. During the first moments of Subject Orchestrator's operation, the MEs information from the EMS Simulator is synchronized to the Slicing blocks, creating the *lowest-level Runtime Slicing Entities* objects. More specifically, the `MFI` and `MLI` objects are created in the NFMF inventories and `CapabilitiesSets` containing appropriately initialized `SlsAttributes` are created in the SLSM inventory (see Subsection 5.4.1 for details). The particular `SlsAttributes` contained in the `CapabilitiesSets` and their values assigned to each MFI reflect their technical features and configurations determined by the Operator during RAN site planning. The values of *Capabilities* at the beginning of the Case Study for all MFIs are listed in Table 12 and commented below.

Table 12. Initial state of MFI *Slicing Entities' Capabilities* and the aggregated *Requirements* posed against them.

MFI *Slicing Entities*		**gNB#A**		**gNB#B**		**gNB#C**			**gNB#D**			
		DU#A.1-5	CU-CP&UP#A	DU#B.1-4	CU-CP&UP#B	DU#C.1-2	DU#C.3-4	CU-CP&UP#C	DU#D.1	DU#D.2-3	DU#D.4-5	CU-CP&UP#D
Current (remaining) *Capabilities* values	coverageAreaTAList	[TA1]		[TA2]		[TA3]	[TA4]	[TA3,TA4]	[TA5]	[TA6]	[TA7]	[TA5,TA6,TA7]
	pLMNInfoList	[X:1, X:3, X:4]		[X:1, X:3, X:4, Y:1, Y:3]					[X:1, X:3, Y:1, Y:3]			
	resourceSharingLevel	SHARED										
	prbPartitionMinRatio	100		100		100			100			
	prbPartitionMaxRatio	100		100		100			100			
	schedulingPriorityDl	[0-15]		[0-15]		[0-15]			[0-15]			
	schedulingPriorityUl	[0-15]		[0-15]		[0-15]			[0-15]			
Aggregated Requirements	pLMNInfoList	-	-	-	-	-	-	-	-	-	-	-
	prbPartitionMinRatio	-		-		-	-		-	-	-	
	prbPartitionMaxRatio	-		-		-	-		-	-	-	
	schedulingPriorityDl	-		-		-	-		-	-	-	
	schedulingPriorityUl	-		-		-	-		-	-	-	

The value of the `trackingAreaTAList` *Capability* of DUs #A.1-5 is equal to [TA1], whereas for DU#D.1 it is [TA5] and for DU#D.4-5 it is [TA7]. The `trackingAreaTAList` *Capability* of DUs is aggregated towards the *Capabilities* of their corresponding CU-CPs and CU-UPs[173]. For example, the CU-CP#D and CU-UP#D exhibit the following value of the `trackingAreaTAList` *Capability*: [TA5, TA6, TA7]. In turn, for CU-CP#B and CU-UP#B = [TA3, TA4]. The value of the `resourceSharingLevel` *Capability* in case of all operated MFIs is always `SHARED`[174].

The information about the Slice types (SST) supported by the MFIs in the particular locations, combined with the IDs of the PLMNs that are allowed to allocate NSSIs of such types, constitutes the value of the `pLMNInfoList` *Capability* of the particular gNB installations. As indicated in Table 12, the value of this *Capability* for all MFIs comprising gNB #A is equal to [X.1, X.3, X.4] which means that PLMN X can allocate Subnets of type eMBB (1), mIoT (3) and V2X (4) in the area covered by gNB#A. According to the agreed `DesignPolicies`, allocation of NSSIs in TA1 is not possible for PLMN Y. Moreover, value "X:4" is present, because Cells managed by gNB#A are operated in 3,5GHz bands, which allows allocation of V2X-typed NSSIs.

[173] See Figure 46 in Subsection 5.5.2.4 for the illustration of *Capabilities aggregation*.

[174] In the PoC, exclusive assignment of *Slicing Entities* to serve a single `SliceProfile` is not supported on the level of the individual *lower-level Slicing Entities* (MFI, MLI). It is supported on the level of an NSSI (see Scenarios 4-5 of the Case Study in Subsections 6.3.4.5 and 6.3.4.6).

In turn, gNB#D's `pLMNInfoList` *Capability* is equal to [X.1, X.3, Y:1, Y:3], since the covered TAs are available for both PLMNs and V2X capabilities are not offered by gNBs operating 2,5GHz Cells.

The initial values of the DUs' lower-level *Capabilities* are specified as well. The `prbPartitionMinRatioDl` and `prbPartitionMaxRatioDl` are equal to 100%, since all the PRBs in the administered RAN domain are dedicated for utilization in sliced networks only. Also, full spectrum of values for configuring the `schedulingPriorityDl` and `schedulingPriorityDl` in range [1-15] is possible.

Among the discussed MFIs' *Capabilities*, the majority exhibits `capabilityConsumptionBehavior` determined as `NO_IMPACT` (indicated in Table 12 with yellow color). This means that their values stay constant regardless of the *Requirements* assigned to them. Only for `prbPartitionMinRatioDl` the `capabilityConsumptionBehavior` is `SUBTRACT` (indicated in Table 12 with blue color), which means that the **value represented by this *Capability* is a crucial resource that decreases with the allocation of new NSSIs**.

While going through the course of the Case Study, **the most important MFIs' *Capabilities* being referred to are**: **`prbPartitionMinRatioDl`**, due to its deductible nature as described above, as well as **`coverageAreaTaList`** and **`pLMNInfoList`**, as they determine the coverage, Slice characteristics and accessibility of a given MFI for PLMNs. The rest of *Capabilities* are less important to be tracked in the PoC, as they are always evaluated as fulfilling the posed *Requirements* with any value (providing it is valid). More specifically, the available `schedulingPriorityDl` and `schedulingPriorityUl` remain the same [1-15] (although assignment of the same priority will cause equal prioritization of traffic for different `SliceProfiles`) and the `prbPartitionMaxRatioDl` is always 100, which satisfies any allocation request. As such, presence of these *Capabilities* is important for appropriate consideration of the MFI *candidates* in the design process, but they do not have major impact on the calculations performed in the particular design activities and, as such, do not need to be tracked.

Taking into account the explanations above, for the sake of brevity and simplification, presentation of *Capabilities* with constant values is omitted in the upcoming, similar tables that are used for tracking the state of *Runtime Slicing Entities* in the course of the Case Study – only the value of MFIs' `prbPartitionMinRatioDl` *Capability* is tracked in those tables and the currently assigned aggregated *Requirements*, as shown in an example of such Table 13 below.

Table 13. Case Study operational objects' tracking table, presenting the state of MFIs and NSSIs *Slicing Entities' Capabilities* and the aggregated *Requirements* assigned to them – initial state.

Initial state of operational objects			MFI *Slicing Entities*							gNB#A		gNB#B		gNB#C			gNB#D			
										DU#A.1-5	CU-CP&UP	DU#B.1-4	CU-CP&UP	DU#C.1-2	DU#C.3-4	CU-CP&UP	DU#D.1	DU#D.2-3	DU#D.4-5	CU-CP&UP
			Current *Capabilities*:						prbPartitionMinRatioDl	100		100		100	100		100	100	100	
			Aggregated Requirements:						prbPartitionMinRatioDl	0		0		0	0		0	0	0	
									prbPartitionMaxRatio	0		0		0	0		0	0	0	
									schedulingPriorityDl	[]		[]		[]	[]		[]	[]	[]	
									schedulingPriorityUl	[]		[]		[]	[]		[]	[]	[]	
									pLMNInfoList	[]										
NSSI *Slicing Entities*	Aggregated *Capabilities*: (total / current)	pLMNInfo List	coverage AreaTAList	resource SharingLevel	prbPartition MinRatioDl	prbPartition MaxRatioDl	scheduling PriorityDl	scheduling PriorityUl	MFIs' assignment to NSSIs and `SliceProfiles`											
-	-	-	-	-	-	-	-	-	none											

Besides the MFI's *Capabilities* and assigned *Requirements*, the *aggregated Capabilities* of NSSIs that will be allocated in the course of the Case Study are presented, the MFIs they utilize and the `SliceProfiles` they are currently assigned with. Tracking the state of *Runtime Slicing Entities*' *Capabilities* in this way allows for verifying whether the ASD PoC mechanisms conduct the design process as expected in the particular Scenarios and Test Cases of the Case Study, which are specified in the following Subsection.

6.3.3. Scenarios and Test Cases specification

The presented Case Study considers the Subject Orchestrator's operation in the described RAN domain within a given, continuous time range. In the considered timeframe, the ASD workflow handles NSSI allocation requests triggered by PLMN X and Y, who construct NSIs with various demands. All the specified Test Cases (TCs) revolve around invoking the `allocateNssi(SliceProfile)` requests with various SLS *Requirements attributes* and their values in the input `SliceProfile`, in varying operational conditions and with different `DesignPolicies`, as specified in Table 14 and commented below.

Table 14. Scenarios and Test Cases specification within the Case Study functional evaluation.

Scenario number and description	Test case ID	Test case description	Input `SliceProfile`	Expected course of the design process	Expected result of the design process	Expected changes in the state of operational objects after NSSI allocation
Scenario 1: greenfield conditions: gNB installations deployed and operational, no NSSIs created yet no `SliceProfiles` exist in the system yet	TC1.rainy.1	invalid `SlsAttribute`	[…] **`uESpeed`** = 200	Allocation workflow interrupted in phase (A) on the syntactic validation activity (step 2)	interrupted	no changes incurred
	TC1.rainy.2	invalid `SlsAttribute` value	[…] `dlLatency` = **`LOW`**			
	TC1.rainy.3	Request from PLMN **Z** (not a Customer)	[…] `pLMNInfoList` = [(**`Z`**:1)]	Allocation workflow interrupted in phase (A) on the semantic validation activity (step 5)		
	TC1.rainy.4	PLMN Y requests NSSI in TA1 (not allowed)	[…] `coverageAreaTAList` = [**`TA1`**, […]]			
	TC1.rainy.5	PLMN Y requests V2X NSSI (not allowed)	[…] `pLMNInfoList` = [(`Y`:**4**)]			
	TC1. sunny	PLMN X requests NSSI for constructing a "X-Premium eMBB" NSI	serviceName="X-Premium-eMBB" pLMNInfoList = [X:1] coverageAreaTAList = TA1-7 resourceSharingLevel = SHARED dlLatency = 50 ulLatency = 50 dlThptPerSliceSubnet = 1000 maxNumberofUEs = 50000	• Selection of `EMBB_NSST` for *Requirements decomposition* into: `prbPartitionMin`/`MaxRatioDl` = 15/80, `schedulingPriorityDl`/`Ul` = 4/7 • Existing NSSI evaluation and selection unsuccessful (no NSSI *candidates*) • `LOW_LEVEL_RAN_NSST` for Subnet design • Evaluation and successful selection of all MFIs *candidates* (full coverage) • *Aggregated Capabilities* of selected candidates `FULFILL` the `SliceProfile`	design of a new NSSI	- NSSI#1 created with all MFIs utilized internally (because full coverage requested) - consumption of 15% PRBs in all DUs and 80% utilization upper limit
Scenario 2: NSSI supporting "X-Premium-eMBB" `SliceProfile` exists	TC2.rainy.X	…	…	…	…	…
	TC2. sunny	PLMN Y requests NSSI for constructing a "Y-Premium eMBB" NSI	serviceName = "Y-Premium-eMBB" pLMNInfoList = [Y:1] coverageAreaTAList = TA2-7 resourceSharingLevel = SHARED dlLatency = 75 ulLatency = 75 dlThptPerSliceSubnet = 600 maxNumberofUEs = 20000	• Selection of `EMBB_NSST` for *Requirements decomposition* into: `prbPartitionMin`/`MaxRatioDl` = 10/60, `schedulingPriorityDl`/`Ul` = 8/9 • Existing NSSI evaluation and selection unsuccessful (NSSI does not offer free PRBs) • `LOW_LEVEL_RAN_NSST` for Subnet design • Evaluation and successful selection of gNB B-D MFIs *candidates* • *Aggregated Capabilities* of selected candidates `FULFILL` the `SliceProfile`	design of a new NSSI	- NSSI#2 created with gNB B-D MFIs utilized internally - consumption of 15% PRBs in all DUs and 60% utilization upper limit
Scenario 3: NSSIs supporting "X-Premium-eMBB" and "Y-Premium-eMBB" `SliceProfiles` exist	TC3.rainy.X	…	…	…	…	…
	TC3. sunny	PLMN Y requests NSSI allocation for transmitting monitoring data from SmartCity sensors	serviceName = "Y-SmartCity-mIoT" pLMNInfoList = [Y:3] coverageAreaTAList = TA3,4,6,7 resourceSharingLevel = SHARED ulThptPerSliceSubnet = 100 maxNumberofUEs = 20000	• Selection of `MIOT_NSST` for *Requirements decomposition* into: `prbPartitionMin`/`MaxRatioDl` = 0/30, `schedulingPriorityDl`/`Ul` = -/11 • Existing NSSI evaluation and selection	reuse of existing NSSI	- `SliceProfile` assigned to one of the previously deployed NSSIs - traffic of the `SliceProfile` scheduled in the NSSI with lower priorities - no dedicated PRBs and upper utilization limit 30%
Scenario 4: NSSIs supporting "X-Premium-eMBB", "Y-Premium-eMBB" and "Y-SmartCity-mIoT" `SliceProfiles` exist	TC4.rainy.X	…	…	…	…	…
	TC4. sunny	PLMN X requests allocation of a non-shared NSSI with stringent QoS requirements to support V2X services on the highways	serviceName = "X-Highways-V2X" pLMNInfoList = [X:4] coverageAreaTAList = TA2-4 resourceSharingLevel = NON_SHARED dlLatency = 10 ulLatency = 10 dlThptPerSliceSubnet = 500 ulThptPerSliceSubnet = 300 maxNumberofUEs = 10000	• Selection of `V2X_NSST` for *Requirements decomposition* into: `prbPartitionMin`/`MaxRatioDl` = 20/80, `schedulingPriorityDl`/`Ul` = 1/1 • Existing NSSI evaluation and selection unsuccessful (NSSIs do not offer free PRBs and are already utilized by other SliceProfiles) • `LOW_LEVEL_RAN_NSST` selected for Subnet design • Evaluation and successful selection of gNB B-C MFIs candidates • *Aggregated Capabilities* of selected candidates `FULFILL` the `SliceProfile`	design of a new NSSI	- NSSI#3 created with gNB B-C MFIs utilized internally (highway coverage) - consumption of 20% PRBs in all gNB B-C DUs and 80% utilization upper limit - no other `SliceProfile` can be assigned to the newly created NSSI#1 (NON_SHARED)
Scenario 5: NSSIs supporting "X-Premium-eMBB", "Y-Premium-eMBB" "Y-SmartCity-mIoT" and "X-Highways-V2X" `SliceProfiles` exist	TC5.rainy.X	…	…	…	…	…
	TC5. sunny	PLMN Y requests allocation of an NSSI for supporting the monitoring of main roads	serviceName = "Y-Roads-Monitoring-mIoT" pLMNInfoList = [Y:3] coverageAreaTAList = TA2-4 resourceSharingLevel = SHARED ulThptPerSliceSubnet = 100 maxNumberofUEs = 30000	• Selection of `MIOT_NSST` for *Requirements decomposition* into: `prbPartitionMin`/`MaxRatioDl` = 0/30, `schedulingPriorityDl`/`Ul` = -/11 • Existing NSSI evaluation and selection	reuse of existing NSSI	- `SliceProfile` assigned to one of the previously deployed NSSIs - traffic of the `SliceProfile` scheduled with lower priorities - no dedicated PRBs and upper utilization limit 30%

All TCs are grouped into 5 Scenarios arranged to verify the operation of the ASD workflow on its main execution paths. Each Scenario represents the operational conditions and the state of operational objects in the Subject Orchestrator's inventories that remain unchanged for the group of TCs executed within it. The group of TCs within each Scenario consists of several "rainy-day" TCs and one "sunny-day" TC. The "rainy-day" TCs exercise the PoC solution's resiliency to invalid inputs and verify correct functioning when the experienced conditions are unfavorable for successful handling of a given NSSI allocation request. As the last one in each Scenario, a "sunny-day" TC is executed, which verifies correct functioning of the PoC solution with an input `SliceProfile` that in the assumed operational conditions is expected to result in a successful allocation of an NSSI. As such, execution of a "sunny-day" TC in a given Scenario **incurs changes in the state of the operational objects**, which moves the Case Study to the next Scenario. Such an organization of the Case Study allows for verifying the correct functioning of the ASD workflow in varying conditions and, at the same time, illustrating the effects brought into the operational state of the objects managed by the Subject Orchestrator as a consequence of applying the design decisions.

A more structured TCs' specification is provided in Table 14, where the following items are specified:

- the environmental conditions of a given Scenario
- ID and brief description of the TC
- `SlsAttributes` and their values comprising the input `SliceProfile`
- expected course of the design process and its expected results
- expected changes incurred in the state of operational objects after NSSI allocation.

It shall be noted that the TCs specified in this Section, which are focused on functional verification, are further supplemented by a large set of TCs executed in the performance evaluation. The TCs described therein, although focused on testing the solution in various load conditions, are verifying the PoC's correct functioning on larger sets of operational input in various arrangements, which contributes to a more thorough functional verification of the elaborated concepts at the same time.

6.3.4. Selected details of Case Study execution and results analysis

Verification of expected functioning of the ASD workflow is analyzed in a "semi-black-box" manner. For each processed allocation request, the ASD workflow produces an operational log that contains details of the executed design activities, their inputs, outputs as well as explanations for the choices of the execution flow in the particular circumstances. In the first place, the final outcome of the end-to-end allocation procedure is checked, because it is the most indicative result of the conducted tests. Then, the log output is analyzed for highlighting selected details of processing in the executed design activities, in order to track and verify the expected functioning of the ASD mechanisms and the changes incurred in the state of the operational objects by applying the design decisions. Due to the fact that the volume of such logs is large, only their selected and abbreviated snippets are presented.

6.3.4.1. "Rainy-day" TCs in Scenario 1

The Case Study begins with Scenario 1, which assumes greenfield conditions – there are no Slices allocated in this part of the Provider's network yet. In these circumstances, several "rainy-day" TCs are executed, which verify correct functioning of the PoC solution for a combination of inputs that should not result in the allocation of an NSSI.

In TC#1.rainy.1-2, the input `SliceProfile` contains syntactic errors in the names and values of the `SlsAttributes`. Such allocation requests are immediately refused during the validation step executed in phase (A) in step 2, without executing any further workflow steps and without incurring any changes in the state of the managed operational objects. For example, for TC#1.rainy.1 the operational log is as follows (numbers in the parentheses indicate the steps of the ASD end-to-end workflow, as defined in Figure 52 in Subsection 5.6.1):

```
(1) Received allocateNssi request with the following SliceProfile:
{
  "serviceName" : "X-unsupported-SLS",
  "pLMNInfoList" : [ "X:1" ]
  "coverageAreaTAList" : [ "TA3", "TA2" ],
  "ueSpeed" : 200
}
(2) Syntactic validation of SliceProfile failed: Unknown SLS attribute: "ueSpeed"
```

In turn, in TC#1.rainy.3-5, a syntactically valid `SliceProfile` is specified. However, it contains `SlsAttributes'` values that are not allowed to be requested. In TC#1.rainy.3, a request from an unknown PLMN Z, which is not an authorized Provider's Customer, is received. In TC#1.rainy.4, PLMN Y attempts to allocate an NSSI on the entire coverage of the RAN domain, including TA1, which according to the `DesignPolicies` is not accessible for PLMN Y. In turn, in TC#1.rainy.5, PLMN Y requests the allocation of a V2X-typed NSSI which, according to the negotiated `DesignPolicies`, is not allowed. The ASD mechanism correctly refuses all such attempts during semantic validation of the request in step 5, interrupting the execution of the ASD workflow without incurring any changes in the state of the operational objects. For example, the operational log for executing TC#1.rainy.5 states:

```
(1) Received allocateNssi request with the following SliceProfile:
{
  "serviceName" : "Y-not-allowed-SST",
  "pLMNInfoList" : [ "Y:4" ]
  "coverageAreaTAList" : [ "TA3", "TA2" ],
  "resourceSharingLevel" : "SHARED",
  "dlLatency" : 50,
  "ulLatency" : 50,
  "maxNumberofUEs" : 25000
}
(3) Loaded DesignPolicies for PLMN 'Y'
         prbUtilizationLimit: 60%
         allowedSsts: [1, 3]
         [..]
(5) Validation of allocation request against DesignPolicies for PLMN 'Y' FAILED: Requested SliceService Type (SST) '4' is not allowed for PLMN 'Y'
```

Similar TCs as described above, which verify correct functioning of the PoC solution in various "rainy-day" conditions, are executed in all remaining Scenarios considered in the Case Study. However, for brevity reasons, the descriptions provided in the upcoming Subsections are focused on the "sunny-day" TCs only, pointing the reader's attention to the most important aspects of the ASD workflow operation in the given circumstances.

6.3.4.2. "Sunny-day" TC in Scenario 1

In TC#1.sunny, NSSI allocation request arrives from PLMN X, who is in the process of creating an NSI that will be dedicated to offer a "Premium-eMBB" Service for its platinum subscribers. The created NSI is meant for general use by the eligible subscribers as an alternative to the "one-size-fits-all" network, by offering increased stability of throughput and guaranteed RTT not exceeding 100ms. The `SliceProfile` passed in the `allocateNssi` request, which expresses these demands, is as follows:

```
(1) Received allocateNssi request with the following SliceProfile:
{
  "serviceName" : "X-Premium-eMBB",
  "pLMNInfoList" : [ "X:1" ]
  "coverageAreaTAList" : [ "TA3", "TA2", "TA1", "TA7", "TA6", "TA5", "TA4" ],
  "resourceSharingLevel" : "SHARED",
  "dlLatency" : 50,
  "ulLatency" : 50,
  "dlThptPerSliceSubnet" : 1000,
  "maxNumberofUEs" : 50000,
}
(3) Loaded DesignPolicies for PLMN 'X'
        prbUtilizationLimit: 80%
        allowedSsts: [3, 1, 4]
        allowedCoverageAreaTAList: [TA7, TA6, TA5, TA4, TA3, TA2, TA1]
        slicingEntitiesSelectionPolicies['Subnet']:
                - selectionPriority: 'DESIGN-TIME';'RUNTIME'
                - requirementsFulfillmentThreshold: 'FULFILLS'
                - candidatesCount: 'SINGLE'
                - existingCandidatesSelectionStrategy: 'LOWEST_REQUIREMENTS_COUNT'
        slicingEntitiesSelectionPolicies['Managed Function']:
                - selectionPriority: 'RUNTIME'
                - requirementsFulfillmentThreshold: 'CONTRIBUTES'
                - candidatesCount: 'MULTIPLE'
                - existingCandidatesSelectionStrategy: 'ALL'
(4) Validation of allocation request against DesignPolicies for PLMN 'X': SUCCESSFUL
```

The expected result of handling such a request, taking into account the currently experienced greenfield conditions, is that the ASD workflow will automatically design a new NSSI that will be composed of all MFIs in the administered RAN domain (coverage of all TAs in the domain is requested). As specified in the last column of Table 14, it is expected that the newly designed NSSI will consume 15% of PRBs in all Cells and its traffic should have priorities: 7 in uplink and 4 in downlink[175].

[175] See Table 7 in Section 5.6.2, where the RDRs for an eMBB-typed Subnet are defined.

Execution of TC#1.sunny in a "black box" manner brings the expected effects – a new NSSI named "NSSI#1" has been deployed and the "X-Premium-eMBB" `SliceProfile` has been assigned to it. From a high-level perspective this is illustrated in Figure 59:

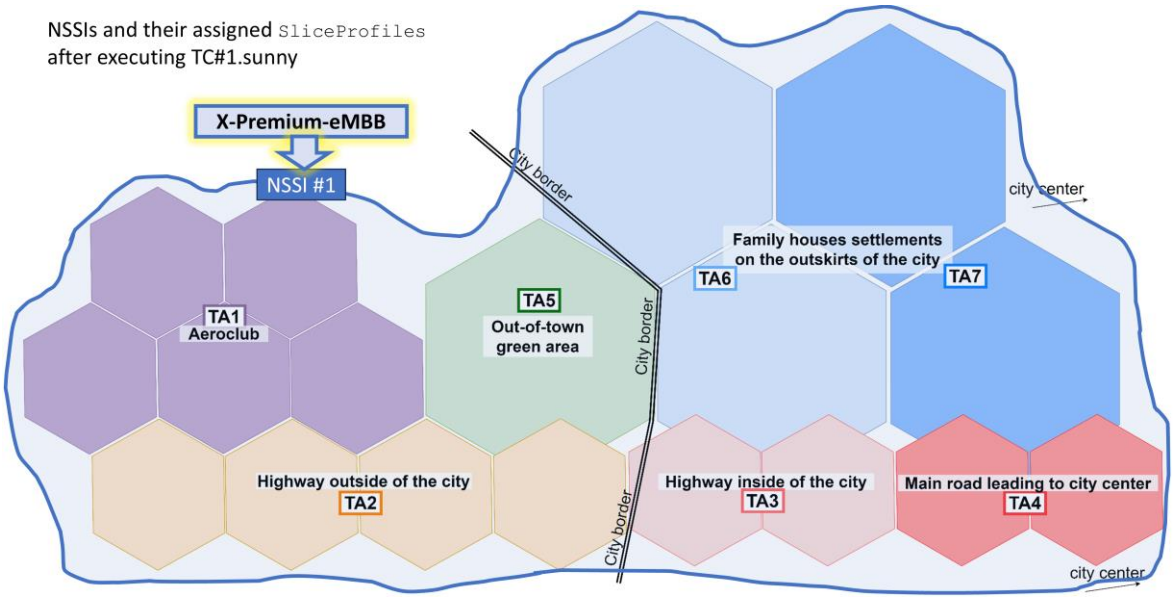


Figure 59. Geographical coverage of the provisioned NSSIs and their assigned `SliceProfiles` after executing TC#1.sunny.

Analysis of the operational log reveals that during TC#1.sunny execution the ASD mechanisms performed, among others, a series of activities commented below:

- In steps 6-7, all available *Slicing Entities* representing Subnet are fetched and evaluated - currently these are only NSSTs, because there are no NSSIs created yet:

```
(6) Fetched candidates representing 'Subnet': [NSST='V2X_NSST', NSST='MIOT_NSST', NSST='EMBB_NSST',
NSST='LOW_LEVEL_RAN_NSST']
(7) Evaluation of candidates for fulfillment of SLS Requirements:
  -> Candidates that cross the evaluationThreshold = 'FULFILLS': ["EMBB_NSST"]
    - 'EMBB_NSST' evaluated as 'FULFILLS' with Capabilities evaluation results:
       dLLatency = '20' -> 'EXCEEDS'
       uLThptPerSliceSubnet = '1000' -> no requirement posed
       uLLatency = '20' -> 'EXCEEDS'
       dLThptPerSliceSubnet = '1000' -> 'FULFILLS'
       maxNumberofUEs = '1000000' -> 'EXCEEDS'
       pLMNInfoList = '[Y:1, X:1]' -> 'EXCEEDS'
       coverageAreaTAList ='[X:TA7,X:TA6,Y:TA6,Y:TA7,Y:TA2,Y:TA3,Y:TA4,Y:TA5,X:TA5,X:TA4,X:TA3,X:TA2,X:TA1]' -> 'EXCEEDS'
       resourceSharingLevel = 'SHARED' -> 'FULFILLS'
    -> Candidates that DO NOT cross the evaluationThreshold = 'FULFILLS': ["MIOT_NSST", "LOW_LEVEL_RAN_NSST", "V2X_NSST"]
       [...]
```

- In steps 8-13, `NSST_EMBB` was selected as the only *candidate* that met the posed *Requirements*:

```
(8) Candidates selection according to slicingEntitiesSelectionPolicies['Subnet']:
    - candidatesCount: 'SINGLE'
    - candidatesPriority: 'DESIGN-TIME';'RUNTIME'
    - evaluationThreshold: 'FULFILLS'
    - runtimeCandidatesPriority: 'LOWEST_REQUIREMENTS_COUNT'
(13) Selected candidate: NSST='EMBB_NSST'
```

- Afterwards, the `EMBB_NSST`'s structure was parsed (step 14) and the high-level SLS *Requirements* were decomposed according to the `RDRs` contained in `EMBB_NSST` [176]. As a result, the lower-level SLS *Requirements* have been produced (step 15):

```
(14) Fetching and parsing NSST 'EMBB_NSST'
       - Parsed topology of Abstract Nodes:
              ['Subnet']
       - Parsed Requirements Decomposition Rules (RDR)
              ['Subnet'].dLLatency:
                     -> ['Subnet'].prbPartitionMinRatioDl: DL_LATENCY_TO_PRB_MIN(eMBB)
                     -> ['Subnet'].prbPartitionMaxRatioDl: DL_LATENCY_TO_PRB_MAX(eMBB)
                     -> ['Subnet'].schedulingPriorityDl: DL_LATENCY_TO_SCHEDULING_PRIORITY_DL(eMBB)
              ['Subnet'].dLThptPerSliceSubnet:
                     -> ['Subnet'].prbPartitionMinRatioDl: DL_THPT_PER_SLICE_SUBNET_TO_PRB_MIN(eMBB)
                     -> ['Subnet'].prbPartitionMaxRatioDl: DL_THPT_PER_SLICE_SUBNET_TO_PRB_MAX(eMBB)
                     -> ['Subnet'].schedulingPriorityDl: DL_THPT_PER_SLICE_SUBNET_TO_SCHEDULING_PRIORITY_DL(eMBB)
              ['Subnet'].uLLatency:
                     -> ['Subnet'].schedulingPriorityUl: UL_LATENCY_TO_SCHEDULING_PRIORITY_UL(eMBB)
              ['Subnet'].maxNumberofUEs:
                     -> ['Subnet'].prbPartitionMinRatioDl: MAX_NUMBER_OF_UES_TO_PRB_MIN(eMBB)
                     -> ['Subnet'].prbPartitionMaxRatioDl: MAX_NUMBER_OF_UES_TO_PRB_MAX(eMBB)

(15) Decomposing SLS Requirements according to RDRs:
```

[176] See Table 7 in Subsection 5.6.2, which specifies the `RDRs` for eMBB-typed Subnets.

```
        ['Subnet'].dLThptPerSliceSubnet = 1000
        ['Subnet'].maxNumberofUEs = 50000
        ['Subnet'].dLLatency = 50
        ['Subnet'].uLLatency = 50

        -> Decomposed SLS Requirements:
                ['Subnet'].schedulingPriorityUl = 7
                ['Subnet'].schedulingPriorityDl = 4
                ['Subnet'].prbPartitionMaxRatioDl = 80
                ['Subnet'].prbPartitionMinRatioDl = 15
```

- Due to the fact that there is a single `AbstractNode` representing Subnet in the parsed `EMBB_NSST`, the workflow proceeded with step 16 towards recursive execution of step 6. The recursive execution of steps 6-13 resulted in the selection of the `LOW_LEVEL_RAN_NSST` as it was the only available *Slicing Entity candidate* representing Subnet that met the posed lower-level SLS *Requirements*:

```
(6) Fetched candidates representing 'Subnet': [NSST='V2X_NSST', NSST='MIOT_NSST', NSST='EMBB_NSST',
NSST='LOW_LEVEL_RAN_NSST']
(7) Evaluation of candidates for fulfillment of SLS Requirements:
    schedulingPriorityUl = '7'
    coverageAreaTAList = '[X:TA7, X:TA6, X:TA5, X:TA4, X:TA3, X:TA2, X:TA1]'
    pLMNInfoList = '[X:1]'
    schedulingPriorityDl = '4'
    prbPartitionMaxRatioDl = '80'
    resourceSharingLevel = 'SHARED'
    prbPartitionMinRatioDl = '15'

  -> Candidates that cross the evaluationThreshold = 'FULFILLS': ["LOW_LEVEL_RAN_NSST"]
    - 'LOW_LEVEL_RAN_NSST' evaluated as 'FULFILLS' with Capabilities evaluation results:
       schedulingPriorityUl = '[1-15]' -> 'EXCEEDS'
       prbPartitionMinRatioDl = '100' -> 'EXCEEDS'
       pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]' -> 'EXCEEDS'
       schedulingPriorityDl = '[1-15]' -> 'EXCEEDS'
       coverageAreaTAList ='[X:TA7,X:TA6,Y:TA6,Y:TA7,Y:TA2,Y:TA3,Y:TA4,Y:TA5,X:TA5,X:TA4,X:TA3,X:TA2,X:TA1]'-> 'EXCEEDS'
       prbPartitionMaxRatioDl = '100' -> 'EXCEEDS'
       resourceSharingLevel = 'SHARED' -> 'FULFILLS'
  -> Candidates that DO NOT cross the evaluationThreshold = 'FULFILLS': ["EMBB_NSST", "MIOT_NSST", "V2X_NSST"]
      [...]
```

- In the next steps, the `LOW_LEVEL_RAN_NSST` was parsed, determining the MF and ML `AbstractNodes` that dictate the abstract structure of a Subnet to be designed and the `RDRs` that transform the input SLS *Requirements* against the whole subnet into `RequirementsSets` to be posed against the *candidates* for substituting the particular `AbstractNodes`:

```
(14) Fetching and parsing NSST 'LOW_LEVEL_RAN_NSST'
    - Parsed topology of Abstract Nodes:
            ['DU'] --- (F1-C) --- ['CU-CP'] --- (N2) ---
                   \                 |
                    \               (E1)
                     \                |
                      - (F1-U) --- ['CU-UP'] --- (N3) ---
    - Parsed Requirements Decomposition Rules (RDR)
            ['Subnet'].coverageAreaTAList:
                    -> ['CU-CP'].coverageAreaTAList: COPY
                    -> ['CU-UP'].coverageAreaTAList: COPY
                    -> ['DU'].coverageAreaTAList: COPY
            ['Subnet'].pLMNInfoList:
                    -> ['CU-CP'].pLMNInfoList: COPY
                    -> ['CU-UP'].pLMNInfoList: COPY
                    -> ['DU'].pLMNInfoList: COPY
            [NetworkSliceSubnet].prbPartitionMinRatioDl:
                    -> ['DU'].prbPartitionMinRatioDl: COPY
            [NetworkSliceSubnet].prbPartitionMaxRatioDl:
                    -> ['DU'].prbPartitionMaxRatioDl: COPY
            [NetworkSliceSubnet].schedulingPriorityDl:
                    -> ['DU'].schedulingPriorityDl: COPY
            [NetworkSliceSubnet].schedulingPriorityUl:
                    -> ['DU'].schedulingPriorityUl: COPY
(15) Decomposing SLS Requirements according to RDRs:
    -> Decomposed SLS Requirements:
        ['CUCP'].coverageAreaTAList = '[X:TA7, X:TA6, X:TA5, X:TA4, X:TA3, X:TA2, X:TA1]'
        ['CUCP'].pLMNInfoList = '[X:1]'
        ['CUUP'].coverageAreaTAList = '[X:TA7, X:TA6, X:TA5, X:TA4, X:TA3, X:TA2, X:TA1]'
        ['CUUP'].pLMNInfoList = '[X:1]'
        ['DU'].schedulingPriorityUl = '7'
        ['DU'].coverageAreaTAList = '[X:TA7, X:TA6, X:TA5, X:TA4, X:TA3, X:TA2, X:TA1]'
        ['DU'].pLMNInfoList = '[X:1]'
        ['DU'].schedulingPriorityDl = '4'
        ['DU'].prbPartitionMaxRatioDl = '80'
        ['DU'].prbPartitionMinRatioDl = '15'
(17) Decomposed SLS Requirements pertain to MF and ML Abstract Nodes - designing new NSSI based on 'LOW_LEVEL_RAN_NSST'
```

- The MF and ML `AbstractNodes` and decomposed `RequirementsSets` were utilized in steps 18-23 for fetching, evaluating and selecting the appropriate MFI&MLI *candidates* (due to large volume of the log, its content is shortened with […]):

```
(18) Fetching viable MFI candidates for each MF Abstract Node (with MLIs matching the ML Abstract Nodes topology)...
(19) Evaluation of candidates for fulfillment of SLS Requirements:
   ['CUUP']:
   -> Candidates that cross the evaluationThreshold = 'CONTRIBUTES': ["CU_UP_D", "CU_UP_A", "CU_UP_B", "CU_UP_C"]
   - 'CU_UP_C' evaluated as 'CONTRIBUTES' with Capabilities evaluation results:
        pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]' -> 'EXCEEDS'
        coverageAreaTAList = '[Y:TA3, Y:TA4, X:TA4, X:TA3]' -> 'CONTRIBUTES'
        [...]
   ['CUCP']:
   -> Candidates that cross the evaluationThreshold = 'CONTRIBUTES': ["CU_CP_A", "CU_CP_D", "CU_CP_B", "CU_CP_C"]
```

```
   - 'CU_CP_A' evaluated as 'CONTRIBUTES' with Capabilities evaluation results:
         pLMNInfoList = '[X:4, X:3, X:1]' -> 'EXCEEDS'
         coverageAreaTAList = '[X:TA1]' -> 'CONTRIBUTES'
         [...]
   ['DU']:
   -> Candidates that cross the evaluationThreshold = 'CONTRIBUTES': ["DU_C2", "DU_B1", "DU_D3", "DU_C3", "DU_B2", "DU_A1",
"DU_D4", "DU_A2", "DU_C4", "DU_D5", "DU_B3", "DU_A3", "DU_B4", "DU_A4", "DU_A5", "DU_D1", "DU_C1", "DU_D2"]
   - 'DU_A3' evaluated as 'CONTRIBUTES' with Capabilities evaluation results:
         prbPartitionMinRatioDl = '100' -> 'EXCEEDS'
         schedulingPriorityUl = '[1-15]' -> 'EXCEEDS'
         schedulingPriorityDl = '[1-15]' -> 'EXCEEDS'
         pLMNInfoList = '[X:4, X:3, X:1]' -> 'EXCEEDS'
         coverageAreaTAList = '[X:TA1]' -> 'CONTRIBUTES'
         prbPartitionMaxRatioDl = '100' -> 'EXCEEDS'
         [...]
   - 'DU_D3' evaluated as 'CONTRIBUTES' with Capabilities evaluation results:
         prbPartitionMinRatioDl = '100' -> 'EXCEEDS'
         schedulingPriorityUl = '[1-15]' -> 'EXCEEDS'
         schedulingPriorityDl = '[1-15]' -> 'EXCEEDS'
         pLMNInfoList = '[X:3, Y:1, Y:3, X:1]' -> 'EXCEEDS'
         coverageAreaTAList = '[X:TA6, Y:TA6]' -> 'CONTRIBUTES'
         prbPartitionMaxRatioDl = '100' -> 'EXCEEDS'
(22) Evaluation summary: there are viable candidates for each Abstract Node – proceeding with selection
(23) Candidates selection according to slicingEntitiesSelectionPolicies['Managed Function']:
       - candidatesCount: 'MULTIPLE'
       - candidatesPriority: 'RUNTIME'
       - evaluationThreshold: 'CONTRIBUTES'
       - runtimeCandidatesPriority: 'ALL'
   -> Selected MULTIPLE RUNTIME candidates that ALL CONTRIBUTE to Requirements fulfillment for each Abstract Node
```

- In step 24, the *Aggregated Capabilities* of all selected MFI *candidates* perceived as a group were calculated and verified for the satisfaction of the `RequirementsSet` against the entire Subnet:

```
(24) Calculating Aggregated Capabilities of selected candidates:
       schedulingPriorityUl = '[1-15]'
       coverageAreaTAList = '[X:TA7,X:TA6,Y:TA6,Y:TA7,Y:TA2,Y:TA3,Y:TA4,Y:TA5,X:TA5,X:TA4,X:TA3,X:TA2,X:TA1]'
       pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]'
       schedulingPriorityDl = '[1-15]'
       prbPartitionMaxRatioDl = '100'
       prbPartitionMinRatioDl = '100'
(25) Evaluating Aggregated Capabilities against the input SliceProfile
(28) Aggregated Capabilities of selected candidates FULFILL the SliceProfile requirements
```

- Successful evaluation in steps 25-28 triggered the generation of an `InstanceDescriptor`, in step 29. The produced `InstanceDescriptor` contains the specification of determined MFIs to be utilized as internals of the designed NSSI. The share of their *Capabilities* to be consumed for supporting the input `SliceProfile` is produced as well, taken as the values of the decomposed *Requirements* in step 15 consuming the *Capabilities* of the internal *Slicing Entities*:

```
 (27) Subnet InstanceDescriptor generation based on design process decisions:
       - name = 'NSSI#1'
       - description = 'created by ASD PoC'
       - nsstId = 'LOW_LEVEL_RAN_NSST'
       - aggregatedSubnetCapabilities:
              schedulingPriorityUl = '[1-15]'
              prbPartitionMinRatioDl = '15'
              schedulingPriorityDl = '[1-15]'
              prbPartitionMaxRatioDl = '80'
              resourceSharingLevel = 'SHARED'
              coverageAreaTAList = '[X:TA7,X:TA6,Y:TA6,Y:TA7,Y:TA2,Y:TA3,Y:TA4,Y:TA5,X:TA5,X:TA4,X:TA3,X:TA2,X:TA1]'
              pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]'
       - assignedSliceProfile:
              schedulingPriorityUl = '7'
              coverageAreaTAList = '[X:TA7, X:TA6, X:TA5, X:TA4, X:TA3, X:TA2, X:TA1]'
              pLMNInfoList = '[X:1]'
              schedulingPriorityDl = '4'
              prbPartitionMaxRatioDl = '80'
              resourceSharingLevel = 'SHARED'
              prbPartitionMinRatioDl = '15'
       - internalsAndTheirConsumedCapabilitiesShare:
             'CU_CP_A':
                  coverageAreaTAList = '[X:TA1]'
                  pLMNInfoList = '[X:1]'
                  resourceSharingLevel = 'SHARED'
             'CU_CP_B':
                  coverageAreaTAList = '[X:TA2]'
                  pLMNInfoList = '[X:1]'
                  resourceSharingLevel = 'SHARED'
             'CU_CP_C':
                  coverageAreaTAList = '[X:TA4, X:TA3]'
                  pLMNInfoList = '[X:1]'
                  resourceSharingLevel = 'SHARED'
             'CU_CP_D':
                  coverageAreaTAList = '[X:TA7, X:TA5, X:TA6]'
                  pLMNInfoList = '[X:1]'
                  resourceSharingLevel = 'SHARED'

             'CU_UP_A':
                  coverageAreaTAList = '[X:TA1]'
                  pLMNInfoList = '[X:1]'
                  resourceSharingLevel = 'SHARED'

                  [...]

             'DU_A1':
                  schedulingPriorityUl = '7'
                  coverageAreaTAList = '[X:TA1]'
                  pLMNInfoList = '[X:1]'
                  schedulingPriorityDl = '4'
```

```
                    prbPartitionMaxRatioDl = '80'
                    prbPartitionMinRatioDl = '15'
                    [...]
               'DU_A2':
                    schedulingPriorityUl = '7'
                    coverageAreaTAList = '[X:TA1]'
                    pLMNInfoList = '[X:1]'
                    schedulingPriorityDl = '4'
                    prbPartitionMaxRatioDl = '80'
                    prbPartitionMinRatioDl = '15'
                    [...]
```

- Then, phase (D) "New NSSI provisioning" was executed in step 30, based on the delivered `InstanceDescriptor`. Afterwards, phase (E) "NSSI re-configuration for `SliceProfile` allocation" procedures were executed in step 31, based on the specification of *Capabilities* to be allocated for supporting the input `SliceProfile`. Their successful completion resulted in the deployment of a new NSSI (named "NSSI#1") with the input `SliceProfile` assigned to it. That ended the execution of the end-to-end allocation workflow:

```
======= (PHASE D) New NSSI provisioning =======
(30) Executing "New NSSI provisioning" based on generated InstanceDescriptor... COMPLETED
======= (PHASE E) NSSI re-configuration for SliceProfile allocation ========
(31) Executing "NSSI re-configuration for SliceProfile allocation" based on Capabilities consumption spec... COMPLETED
======= End-to-end NSSI allocation workflow COMPLETED SUCCESSFULLY ======
```

Table 15 juxtaposes the values of *Capabilities* exposed by the *Runtime Slicing Entities* (i.e. the already existing MFIs and the newly created NSSI#1) after finishing the entire allocation workflow. Black font represents the *Capabilities/aggregated Requirements* values before TC#1.sunny execution. Green font represents the currently available value (after TC#1.sunny execution), which is calculated by SLSM by taking into account the value of the currently assigned set of SLS *Requirements* (marked in red) aggregated with all the already posed *Requirements* incurred by the already assigned `SliceProfiles` (black font). Bottom-right corner of Table 15 presents a schematic overview of the assignment of MFIs to NSSI#1 as well as the "X-Premium-eMBB" `SliceProfile` assignment to NSSI#1 and its underlying MFIs.

Table 15. Case Study operational objects' tracking table – state after TC#1.sunny execution.

State of operational objects after execution of TC#1.sunny			MFI *Slicing Entities*						gNB #A		gNB #B		gNB #C		gNB #D		
									DU#A.1-5	CU-CP&UP	DU#B.1-4	CU-CP&UP	DU#C.1-2	DU#C.3-4	CU-CP&UP	DU#D.1	DU#D.2-3	DU#D.4-5	CU-CP&UP
			Current *Capabilities*:					prbPartitionMinRatioDl	100-15=85		100-15=85		100-15=85			100-15=85			
			Aggregated Requirements:					prbPartitionMinRatioDl	0+15=15		0+15=15		0+15=15			0+15=15			
								prbPartitionMaxRatio	max(0;80)=80		max(0;80)=80		max(0;80)=80			max(0;80)=80			
								schedulingPriorityDl	[]+[4]=[4]		[]+[4]=[4]		[]+[4]=[4]			[]+[4]=[4]			
								schedulingPriorityUl	[]+[7]=[7]		[]+[7]=[7]		[]+[7]=[7]			[]+[7]=[7]			
								pLMNInfoList	[]+[X:1]=[X:1]										
NSSI *Slicing Entities*			pLMNInfo List	coverage AreaTAList	resource SharingLevel	prbPartition MinRatioDl	prbPartition MaxRatioDl	scheduling PriorityDl	scheduling PriorityUl	MFIs' assignment to NSSIs and `SliceProfiles`									
NSSI #1	Aggregated *Capabilities*: (total / current)		[X:1,3; Y:1,3]	TA1-7	SHARED	15 / 0	80 / 80	[1-15]	[1-15]	NSSI #1									
	Slice Profiles	X-Premium-eMBB	[X:1]	TA1-7	SHARED	15	80	4	7	X-Premium-eMBB									

Analysis of the consumed values of MFIs' *Capabilities* and the *aggregated Capabilities* exposed by NSSI#1 confirm that the ASD workflow automatically **performed a correct design process that followed the predefined `DesignPolicies`.**

It is worth noting that the specification of the new NSSI#1 in the generated `InstanceDescriptor` assumes consuming 15% share of the `prbPartitionMinRatioDl` *Capability* in the underlying DU MFIs, while currently all of them offer 100%. It is the expected result of the design process configured by the Operator to create fine-grained NSSIs designed to closely match the posed *Requirements*. According to a different strategy that could be implemented, the designed NSSI could consume a higher share of deductible *Capabilities* (e.g. defined by an excess percentage) in order for such an NSSI to contain "spare" *Capabilities*. This could result in the potential reuse of such NSSI by a larger number of NSIs (more `SliceProfiles` attached). However, the variant of allocating NSSIs with "spare" *Capabilities* is not considered in the Case Study, but considered as an extension of the PoC.

6.3.4.3. Scenario 2

TCs in Scenario 2 are executed in the operational conditions incurred by TC#1.sunny, thus existence of NSSI#1 assigned with the "X-Premium-eMBB" `SliceProfile` is assumed. This time, the NSSI allocation request is invoked by PLMN Y, who aims at offering a similar "Y-Premium-eMBB" Service for its clients, but with slightly less stringent QoS expectations.

Description of the "rainy-day" TCs executed in the current Scenario, which check the resiliency of the PoC solution to various corner cases, is omitted for brevity reasons. Also, only the meaningful content of the operational log from the execution of TC#2.sunny is presented, omitting fragments that are similar to the processing that has already been commented in Scenario 1:

- In TC#2.sunny, the `SliceProfile` passed by PLMN Y is as follows:

```
======= (PHASE A) Allocation request analysis ========
(1) Received allocateNssi request with the following SliceProfile:
{
  "serviceName" : "Y-Premium-eMBB",
  "pLMNInfoList" : [ "Y:1" ]
  "coverageAreaTAList" : [ "TA3", "TA2", "TA7", "TA6", "TA5", "TA4" ],
  "resourceSharingLevel" : "SHARED",
  "dlLatency" : 75,
  "ulLatency" : 75,
  "dlThptPerSliceSubnet" : 600,
  "maxNumberofUEs" : 20000,
}
```

- In the execution of initial steps 6-16, selection of the `EMBB_NSST` "translation NSST" and decomposition of high-level SLS *Requirements* were performed similarly as described in Scenario 1, leading to the recursive execution of step 6.
- In the recursive execution of steps 6-17, the already existing NSSI#1 was considered as a preferred *candidate* to be reused, due to the policy `candidatesPriority` set to `RUNTIME;DESIGN-TIME` for PLMN Y. NSSI#1 was evaluated to cover a superset of the requested TAs and offered satisfactory values of almost all *Capabilities*:

```
(6) Fetched candidates representing 'Subnet': [NSSI='NSSI#1', NSST='LOW_LEVEL_RAN_NSST', NSST='MIOT_NSST', NSST='V2X_NSST',
NSST='EMBB_NSST']
(7) Evaluation of candidates for fulfillment of SLS Requirements:
        schedulingPriorityUl: '9'
        prbPartitionMinRatioDl: '10'
        schedulingPriorityDl: '8'
        coverageAreaTAList: '[Y:TA2, Y:TA3, Y:TA4, Y:TA5, Y:TA6, Y:TA7]'
        pLMNInfoList: '[Y:1]'
        resourceSharingLevel: 'SHARED'
        prbPartitionMaxRatioDl: '60'
   -> Candidates that cross the evaluationThreshold = 'FULFILLS': ["LOW_LEVEL_RAN_NSST"]
        [...]
   -> Candidates that DO NOT cross the evaluationThreshold = 'FULFILLS': ["EMBB_NSST", "MIOT_NSST", "NSSI#1", "V2X_NSST"]
     - 'NSSI#1' evaluated as 'DOES_NOT_CONTRIBUTE' with Capabilities evaluation results:
       pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]' -> 'EXCEEDS'
       schedulingPriorityUl = '[1-15]' -> 'EXCEEDS'
       schedulingPriorityDl = '[1-15]' -> 'EXCEEDS'
       prbPartitionMaxRatioDl = '80' -> 'EXCEEDS'
       prbPartitionMinRatioDl = '0' -> 'DOES_NOT_CONTRIBUTE'
       resourceSharingLevel = 'SHARED' -> 'FULFILLS'
       coverageAreaTAList = '[X:TA7,X:TA6,Y:TA6,Y:TA7,Y:TA2,Y:TA3,Y:TA4,Y:TA5,X:TA5,X:TA4,X:TA3,X:TA2,X:TA1]' -> 'EXCEEDS'
         [...]
```

- However, with its fully exceeded `prbPartitionMinRatioDl` *Capability* equal to 0, NSSI#1 did not meet the *Requirement* value 10%, which was needed for the satisfaction of the considered "Y-Premium-eMBB" `SliceProfile`. Thus, "NSSI#1" was evaluated as not fulfilling the posed *Requirements* and, as such, not considered for selection.
- Due to the fact that there were no other existing NSSI *candidates*, the `LOW_LEVEL_RAN_NSST` was selected as the only *Slicing Entity* that crossed the `evaluationThreshold` = `FULFILLS`. The `LOW_LEVEL_RAN_NSST` was utilized in steps 18-29 for designing a new NSSI. The design process was performed similarly as described in Scenario 1, but the MFIs comprising gNB#A did not fulfill the `coverageAreaTAList` and `pLMNInfoList` *Requirements* and the DUs of gNB#A do not offer the demanded *Capabilities* in the requested coverage area:

```
(13) Selected candidate: NSST='LOW_LEVEL_RAN_NSST'
(17) Decomposed SLS Requirements pertain to MF and ML Abstract Nodes - designing new NSSI based on NSST 'LOW_LEVEL_RAN_NSST'
======= (PHASE C) Attempt designing new NSSI =======
(18) Fetching viable MFI candidates for each MF Abstract Node (with MLIs matching the ML Abstract Nodes topology)...
(19) Evaluation of candidates for fulfillment of SLS Requirements:
   ['CUCP']:
     -> Candidates that cross the evaluationThreshold = 'CONTRIBUTES': ["CU_UP_D", "CU_UP_B", "CU_UP_C"]
          [...]
     -> Candidates that DO NOT cross the evaluationThreshold = 'CONTRIBUTES': ["CU_UP_A"]
     - 'CU_UP_A' evaluated as 'DOES_NOT_CONTRIBUTE' with Capabilities evaluation results:
       pLMNInfoList = '[X:4, X:3, X:1]' -> 'DOES_NOT_CONTRIBUTE'
       coverageAreaTAList = '[X:TA1]' -> 'DOES_NOT_CONTRIBUTE'
   ['CUUP']:
     -> Candidates that cross the evaluationThreshold = 'CONTRIBUTES': ["CU_CP_D", "CU_CP_B", "CU_CP_C"]
          [...]
     -> Candidates that DO NOT cross the evaluationThreshold = 'CONTRIBUTES': ["CU_UP_A"]
     - 'CU_CP_A' evaluated as 'DOES_NOT_CONTRIBUTE' with Capabilities evaluation results:
       pLMNInfoList = '[X:4, X:3, X:1]' -> 'DOES_NOT_CONTRIBUTE'
       coverageAreaTAList = '[X:TA1]' -> 'DOES_NOT_CONTRIBUTE'
   ['DU']:
     -> Candidates that cross the evaluationThreshold = 'CONTRIBUTES': ["DU_C2", "DU_B1", "DU_D3", "DU_C3", "DU_B2", "DU_D4",
"DU_C4", "DU_D5", "DU_B3", "DU_B4", "DU_D1", "DU_C1", "DU_D2"]
          [...]
```

```
      -> Candidates that DO NOT cross the evaluationThreshold = 'CONTRIBUTES': ["DU_A1", "DU_A2", "DU_A3", "DU_A4", "DU_A5"]
      - 'DU_A1' evaluated as 'NO_CAPABILITY' with Capabilities evaluation results:
         pLMNInfoList = '[X:4, X:3, X:1]' -> 'DOES_NOT_CONTRIBUTE'
         coverageAreaTAList = '[X:TA1]' -> 'DOES_NOT_CONTRIBUTE'
      - 'DU_A5' evaluated as 'NO_CAPABILITY' with Capabilities evaluation results:
         pLMNInfoList = '[X:4, X:3, X:1]' -> 'DOES_NOT_CONTRIBUTE'
         coverageAreaTAList = '[X:TA1]' -> 'DOES_NOT_CONTRIBUTE'
(23) Candidates selection according to slicingEntitiesSelectionPolicies['Managed Function']:
     - candidatesCount: 'MULTIPLE'
     - candidatesPriority: 'RUNTIME'
     - evaluationThreshold: 'CONTRIBUTES'
     - runtimeCandidatesPriority: 'ALL'
     -> Selected MULTIPLE RUNTIME candidates that ALL CONTRIBUTE to Requirements fulfillment for each Abstract Node
```

- The evaluation results not crossing the `CONTRIBUTES` threshold excluded all the gNB#A MFIs from being considered in the currently designed NSSI, which was the expected behavior. All the other MFI *candidates* were evaluated to meet the `evaluationThreshold` = `CONTRIBUTES` and, as such, were selected in step 23 to be included in the designed NSSI.
- Computation and verification of *aggregated Capabilities* of the selected *candidates* in steps 24-28 is successful (evaluation score `FULFILLS`), which results in the generation of an `InstanceDescriptor` in step 29 and its utilization for the provisioning of a new NSSI in step 30 (phase D) and its configuration for allocation of the input `SliceProfile` in step 31 (phase E):

```
(24) Calculating Aggregated Capabilities of selected candidates:
        schedulingPriorityUl = '[1-15]'
        prbPartitionMinRatioDl = '85'
        schedulingPriorityDl = '[1-15]'
        coverageAreaTAList = '[Y:TA2, Y:TA3, Y:TA4, Y:TA5, X:TA7, X:TA6, X:TA5, X:TA4, X:TA3, X:TA2, Y:TA6, Y:TA7]'
        pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]'
        prbPartitionMaxRatioDl = '100'
(25) Evaluating Aggregated Capabilities against the input SliceProfile
(28) Aggregated Capabilities of selected candidates satisfy the evaluationThreshold: 'FULFILLS'
(29) Subnet InstanceDescriptor generation based on design process decisions:
     [...]
======= (PHASE D) New NSSI provisioning =======
(30) Executing "New NSSI provisioning" based on generated InstanceDescriptor... COMPLETED
======= (PHASE E) NSSI re-configuration for SliceProfile allocation ========
(31) Executing "NSSI re-configuration for SliceProfile allocation" based on Capabilities consumption spec... COMPLETED
======= End-to-end NSSI allocation workflow COMPLETED SUCCESSFULLY ======
```

After successful completion of the entire workflow, the provisioned NSSIs and `SliceProfiles` assigned to them in the covered RAN domain are schematically depicted in Figure 60.

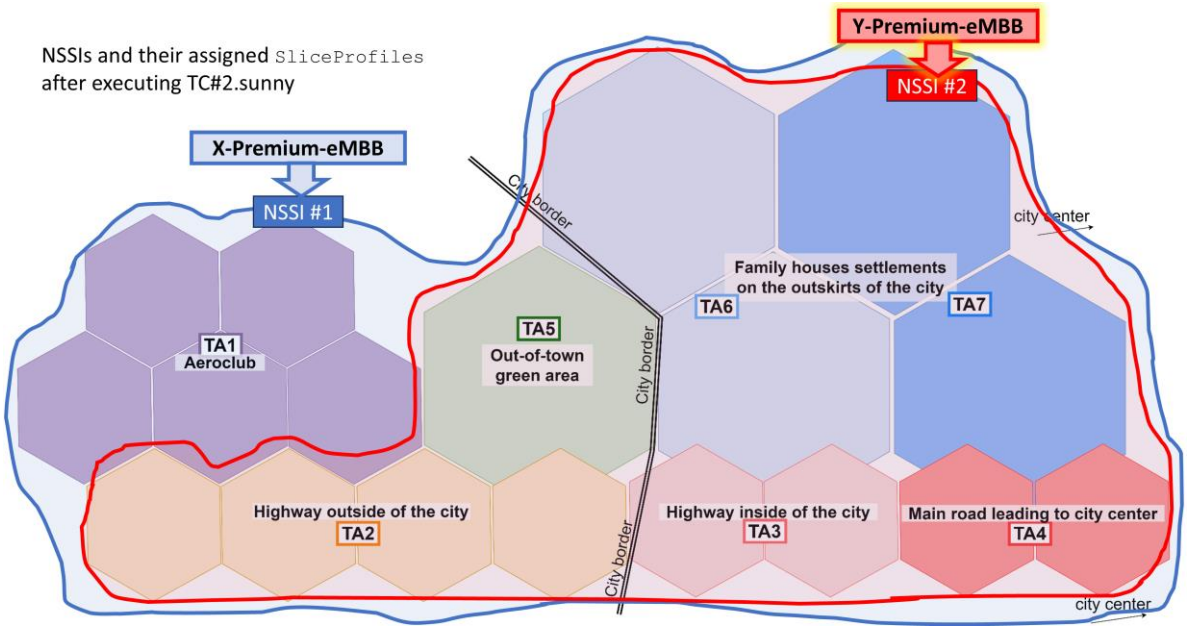


Figure 60. Geographical coverage of the provisioned NSSIs and their assigned `SliceProfiles` after executing TC#2.sunny.

The state of operational objects after TC#2.sunny execution is presented in Table 16. It shall be noted that in the case of gNB#A DUs, their remaining value of the `prbPartitionMinRatioDl` *Capability* did not change and remains at the level of 85%, whereas the value of this *Capability* in the case of other gNB DUs has been deducted and now is equal to 75%.

Table 16. Case Study operational objects' tracking table – state after TC#2.sunny execution.

State of operational objects after execution of TC#2.sunny			MFI *Slicing Entities*								gNB #A		gNB #B		gNB #C			gNB #D			
											DU#A.1-5	CU-CP&UP	DU#B.1-4	CU-CP&UP	DU#C.1-2	DU#C.3-4	CU-CP&UP	DU#D.1	DU#D.2-3	DU#D.4-5	CU-CP&UP
										Current *Capabilities*:	prbPartitionMinRatioDl	85		85-10=75		85-10=75			85-10=75		
										Aggregated Requirements:	prbPartitionMinRatioDl	15		15+10=25		15+10=25			15+10=25		
											prbPartitionMaxRatio	80		max(80;60)=80		max(80;60)=80			max(80;60)=80		
											schedulingPriorityDl	[4]		[4]+[8]=[4,8]		[4]+[8]=[4,8]			[4]+[8]=[4,8]		
											schedulingPriorityUl	[7]		[7]+[9]=[7,9]		[7]+[9]=[7,9]			[7]+[9]=[7,9]		
											pLMNInfoList	[X:1]		[X:1]+[Y:1]=[X:1, Y:1]							
NSSI Slicing Entities			pLMNInfo List	coverage AreaTAList	resource SharingLevel	prbPartition MinRatioDl	prbPartition MaxRatioDl	scheduling PriorityDl	scheduling PriorityUl	MFIs' assignment to NSSIs and SliceProfiles											
NSSI #1	Aggregated *Capabilities*: (total / current)		[X:1,3; Y:1,3]	TA1-7	SHARED	15 / 0	80 / 80	[1-15]	[1-15]	NSSI #1											
	Slice Profiles:	X-Premium-eMBB	[X:1]	TA1-7	SHARED	15	80	4	7	X-Premium-eMBB											
NSSI #2	Aggregated *Capabilities*: (total / current)		[X:1,3; Y:1,3]	TA2-7	SHARED	10 / 0	60 / 60	[1-15]	[1-15]				NSSI #2								
	Slice Profiles:	Y-Premium-eMBB	[Y:1]	TA2-7	SHARED	10	60	8	9				Y-Premium-eMBB								

6.3.4.4. Scenario 3

TC#3.sunny is executed in the environmental conditions incurred by TC#2.sunny, thus existence of NSSI#1 and NSSI#2 with their assigned `SliceProfiles` is assumed. In such circumstances, an `allocateNssi` request from PLMN Y is received, containing `SliceProfile` "Y-SmartCity-mIoT" with SLS *Requirements* of a currently constructed NSI that will be utilized to support SmartCity Services for transmitting the monitoring data from sensors distributed in the city premises. The received `SliceProfile` is as follows:

```
(1) Received allocateNssi request with the following SliceProfile:
{
  "serviceName" : "Y-SmartCity-mIoT",
  "pLMNInfoList" : [ "Y:3" ]
  "coverageAreaTAList" : [ "TA3", "TA7", "TA6", "TA4" ],
  "resourceSharingLevel" : "SHARED",
  "ulThptPerSliceSubnet" : 200,
  "maxNumberofUEs" : 20000,
}
```

By interpreting the obtained SLS *Requirements* it can be stated that the sensors' traffic is not critical to be served in real-time and there are no explicit *Requirements* on the traffic in downlink, thus, it can be handled in the "best-effort" manner. Although a large number of end-devices is expected to be connected in the entire NSI, inside of the considered RAN domain a relatively small number of sensors are expected to be deployed, with `maxNumberofUEs` estimated by PLMN Y to 20 000. The QoS guarantees are not the main reason for PLMN Y to establish an NSI – it is the need to supply an isolated network that will not be accessible to unauthorized UEs.

By taking into account the `DesignPolicies` defined for PLMN Y (in particular, the `candidatesPriority` set to `RUNTIME;DESIGN-TIME`) and due to the fact that no dedicated PRBs are needed for successfully meeting such a `SliceProfile`[177], it is expected that the ASD workflow will decide to reuse one of the already existing NSSI and assign the input `SliceProfile` to it. Execution of TC3.sunny in a "black-box" manner confirms such a behavior - the input "Y-SmartCity-mIoT" `SliceProfile` got assigned to NSSI#2, as depicted in Figure 61:

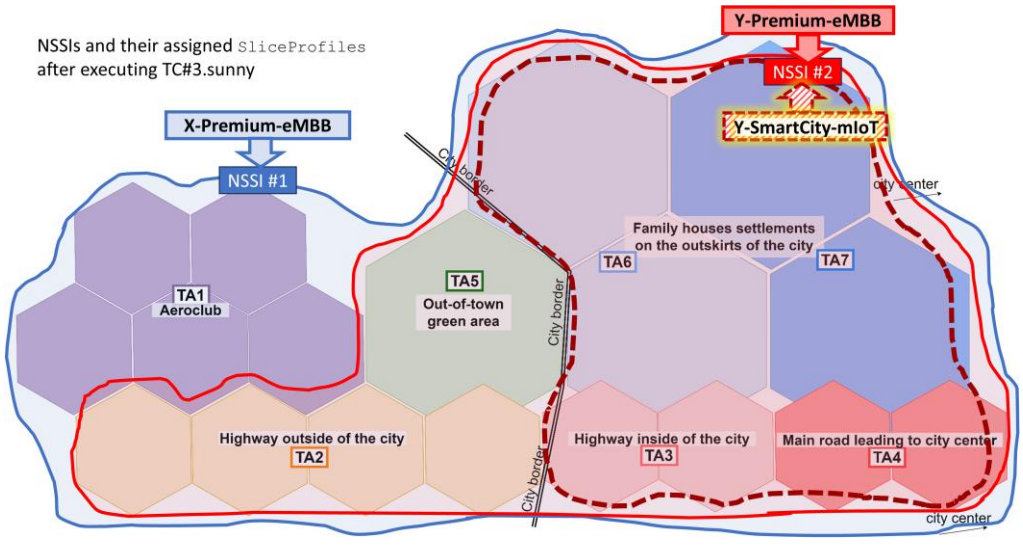


Figure 61. Geographical coverage of the provisioned NSSIs and their assigned `SliceProfiles` after executing TC#3.sunny.

[177] See the RDRs for the mIoT-typed Subnets specified in Table 8 in Subsection 5.6.2.

By analyzing the operational log, it can be checked that the ASD workflow performed, among others, the following processing:

- In the first iteration of steps 6-16, the high-level SLS *Requirements* were decomposed by means of the `MIOT_NSST` "translation NSST", which was the only *candidate* that met the posed SLS *Requirements*. The resulting lower-level SLS *Requirements* were as follows:

```
(15) Decomposing SLS Requirements according to RDRs:
       ['Subnet'].uLThptPerSliceSubnet = 200
       ['Subnet'].maxNumberofUEs = 20000

       -> Decomposed SLS Requirements:
              ['Subnet'].schedulingPriorityUl = 11
              ['Subnet'].prbPartitionMaxRatioDl = 30
              ['Subnet'].prbPartitionMinRatioDl = 0
```

- With the recursive execution of step 6, all *candidates* representing a Subnet were fetched and evaluated in step 7, including all NSSTs and the already existing NSSI#1 and NSSI#2:

```
(6) Fetched candidates representing 'Subnet': [NSSI='NSSI#2', NSSI='NSSI#1', NSST='V2X_NSST', NSST='MIOT_NSST',
NSST='EMBB_NSST', NSST='LOW_LEVEL_RAN_NSST']
(7) Evaluation of candidates for fulfillment of SLS Requirements:
  schedulingPriorityUl = '7'
  coverageAreaTAList = '[Y:TA3, Y:TA4, Y:TA6, Y:TA7]'
  prbPartitionMinRatioDl = '0'
  prbPartitionMaxRatioDl = '30'
  resourceSharingLevel = 'SHARED'
  pLMNInfoList = '[Y:3]' '

  -> Candidates that cross the evaluationThreshold = 'FULFILLS': ["NSSI#2", "NSSI#1", "LOW_LEVEL_RAN_NSST"]
  - 'NSSI#2' evaluated as 'FULFILLS' with Capabilities evaluation results:
     pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]' -> 'EXCEEDS'
     schedulingPriorityUl = '[1-15]' -> 'EXCEEDS'
     schedulingPriorityDl = '[1-15]' -> no requirement posed
     prbPartitionMinRatioDl = '0' -> 'FULFILLS'
     resourceSharingLevel = 'SHARED' -> 'FULFILLS'
     prbPartitionMaxRatioDl = '60' -> 'EXCEEDS'
     coverageAreaTAList = '[Y:TA2, Y:TA3, Y:TA4, Y:TA5, X:TA7, X:TA6, X:TA5, X:TA4, X:TA3, X:TA2, Y:TA6, Y:TA7]' -> 'EXCEEDS'
   - 'NSSI#1' evaluated as 'FULFILLS' with Capabilities evaluation results:
     pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]' -> 'EXCEEDS'
     schedulingPriorityUl = '[1-15]' -> 'EXCEEDS'
     schedulingPriorityDl = '[1-15]' -> no requirement posed
     prbPartitionMaxRatioDl = '80' -> 'EXCEEDS'
     prbPartitionMinRatioDl = '0' -> 'FULFILLS'
     resourceSharingLevel = 'SHARED' -> 'FULFILLS'
     coverageAreaTAList = '[X:TA7,X:TA6,Y:TA6,Y:TA7,Y:TA2,Y:TA3,Y:TA4,Y:TA5,X:TA5,X:TA4,X:TA3,X:TA2,X:TA1]' -> 'EXCEEDS'
   - 'LOW_LEVEL_RAN_NSST' evaluated as 'FULFILLS' with Capabilities evaluation results:
     schedulingPriorityUl = '[1-15]' -> 'EXCEEDS'
     prbPartitionMinRatioDl = '100' -> 'EXCEEDS'
     pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]' -> 'EXCEEDS'
     schedulingPriorityDl = '[1-15]' -> no requirement posed
     coverageAreaTAList = '[X:TA7,X:TA6,Y:TA6,Y:TA7,Y:TA2,Y:TA3,Y:TA4,Y:TA5,X:TA5,X:TA4,X:TA3,X:TA2,X:TA1]' -> 'EXCEEDS'
     prbPartitionMaxRatioDl = '100' -> 'EXCEEDS'
     resourceSharingLevel = 'SHARED' -> 'FULFILLS'
   -> Candidates that DO NOT cross the evaluationThreshold = 'FULFILLS': ["EMBB_NSST", "MIOT_NSST", "V2X_NSST"]
     [...]
```

- There were 3 *candidates* crossing the `evaluationThreshold` for *Slicing Entities* representing Subnets (`FULFILLS`): `LOW_LEVEL_RAN_NSST` and two existing NSSIs: NSSI#1 and NSSI#2. Due to `candidatesPriority` set for PLMN Y to `RUNTIME;DESIGN-TIME`, the selection was performed from the two existing NSSIs
- The `runtimeCandidatesPriority` for PLMN Y was set to `HIGHEST_REQUIREMENTS_COUNT`, but both NSSIs had one `SliceProfile` attached at the moment of this allocation request handling. Thus, the selection was performed randomly from these two *candidates*. In this particular case, NSSI#2 was selected in step 10:

```
(8) Candidates selection according to slicingEntitiesSelectionPolicies['Subnet']:
       - candidatesCount: 'SINGLE'
       - candidatesPriority: 'RUNTIME';'DESIGN-TIME'
       - evaluationThreshold: 'FULFILLS'
       - runtimeCandidatesPriority: 'HIGHEST_REQUIREMENTS_COUNT'
(10) Selected candidate: NSSI='NSSI#2'
```

- Then, in step 11, the *Capabilities'* values of internal *Slicing Entities* that shall be devoted for supporting the input `SliceProfile` have been determined by utilizing the RDRs contained in the NSST corresponding to the selected NSSI#2 (`LOW_LEVEL_RAN_NSST`). Finally, the workflow proceeded with step 12 directly to phase (E), where NSSI#2 and its internal Slicing Entities were re-configured to support the input `SliceProfile` with the decided consumption of *Capabilities*:

```
(11) Determining internal Slicing Entities Capabilities' consumption based on RDRs of 'NSSI#2's NSST:
       - internalsAndTheirConsumedCapabilitiesShare:
            'CU_CP_C':
                 coverageAreaTAList = '[Y:TA3, Y:TA4]'
                 pLMNInfoList = '[Y:3]'
                 resourceSharingLevel = 'SHARED'
            'CU_CP_D':
                 coverageAreaTAList = '[Y:TA7, Y:TA6]'
                 pLMNInfoList = '[Y:3]'
                 resourceSharingLevel = 'SHARED'

            'CU_UP_D':
                 coverageAreaTAList = '[Y:TA7, Y:TA6]'
```

```
                    pLMNInfoList = '[Y:3]'
                    resourceSharingLevel = 'SHARED'
               'CU_UP_C':
                    coverageAreaTAList = '[Y:TA3, Y:TA4]'
                    pLMNInfoList = '[Y:3]'
                    resourceSharingLevel = 'SHARED'

               'DU_C3':
                    schedulingPriorityUl = '7'
                    coverageAreaTAList = '[Y:TA4, Y:TA3]'
                    pLMNInfoList = '[Y:3]'
                    prbPartitionMaxRatioDl = '30'
                    prbPartitionMinRatioDl = '0'
               'DU_D3':
                    schedulingPriorityUl = '7'
                    coverageAreaTAList = '[Y:TA6, Y:TA7]'
                    pLMNInfoList = '[Y:3]'
                    prbPartitionMaxRatioDl = '30'
                    prbPartitionMinRatioDl = '0'
                    [...]

======= (PHASE E) NSSI re-configuration for SliceProfile allocation ========
(12) Executing "NSSI re-configuration for SliceProfile allocation" based on Capabilities consumption spec... COMPLETED
======= End-to-end NSSI allocation workflow COMPLETED SUCCESSFULLY =======
```

After the execution of TC#3.sunny, the state of operational objects is juxtaposed in Table 17.

Table 17. Case Study operational objects' tracking table – state after TC#3.sunny execution.

State of operational objects after execution of TC#3.sunny			MFI *Slicing Entities*							gNB #A		gNB #B		gNB #C			gNB #D			
										DU#A.1-5	CU-CP&UP	DU#B.1-4	CU-CP&UP	DU#C.1-2	DU#C.3-4	CU-CP&UP	DU#D.1	DU#D.2-3	DU#D.4-5	CU-CP&UP
					Current *Capabilities*:				prbPartitionMinRatioDl	85		85		85-0=85			85	85-0=85		
					Aggregated Requirements:				prbPartitionMinRatioDl	15		15		15+0=15			15	15+0=15		
									prbPartitionMaxRatio	80		80		max(80;30)=80			80	max(80;30)=80		
									schedulingPriorityDl	[4]		[4,8]		[4,8]				[4,8]		
									schedulingPriorityUl	[7]		[7,9]		[7,9]+[7]=[7,9,7]			[7,9]	[7,9]+[7]=[7,9,7]		
									pLMNInfoList	[X:1]		[X:1, Y:1]		[X:1,Y:1]+[Y:3]=[X:1, Y:1, Y:3]			[X:1, Y:1]	[X:1,Y:1]+[Y:3]=[X:1, Y:1, Y:3]		
			pLMNInfo List	coverage AreaTAList	resource SharingLevel	prbPartition MinRatioDl	prbPartition MaxRatioDl	scheduling PriorityDl	scheduling PriorityUl	MFIs' assignment to NSSIs and SliceProfiles										
NSSI Slicing Entities	NSSI #1	Aggregated *Capabilities*: (total / current)	[X:1,3; Y:1,3]	TA1-7	SHARED	15 / 0	80 / 80	[1-15]	[1-15]	NSSI #1										
		Slice Profiles: X-Premium-eMBB	[X:1]	TA1-7	SHARED	15	80	4	7	X-Premium-eMBB										
	NSSI #2	Aggregated *Capabilities*: (total / current)	[X:1,3; Y:1,3]	TA2-7	SHARED	10 / 0	60 / 60	[1-15]	[1-15]			NSSI #2								
		Slice Profiles: Y-Premium-eMBB	[Y:1]	TA2-7	SHARED	10	60	8	9			Y-Premium-eMBB								
		Slice Profiles: Y-SmartCity-mIoT	[Y:3]	TA3-4, 6-7	SHARED	0	30	-	11					Y-SmartCity-mIoT				Y-SmartCity-mIoT		

Several decisions made by the ASD mechanisms are worth commenting:

- Traffic of the newly assigned "Y-SmartCity-mIoT" NSI will be transmitted within NSSI#2 with lower priority than the traffic of "Y-Premium-eMBB" NSI and without dedicated occupation of PRBs (`prbPartitionMinRatioDl` *Requirement* = 0). Thus, the traffic of the "Y-SmartCity-mIoT" NSI will be transmitted only if there are currently free, non-dedicated PRBs with the upper limit of 30% utilization.
- NSSI#2, to which the "Y-SmartCity-mIoT" NSI was assigned, covers a superset of TAs with respect to the area requested in the input `SliceProfile`. Thus, although it may seem that the subscribers could connect to the "Y-SmartCity-mIoT" NSI for example in TA2 or TA5, access to the NSI will be blocked by the AMF, which decides on admitting the UE connection to the particular NSI,

6.3.4.5. Scenario 4

TC#4.sunny is executed in the environmental conditions incurred by TC#3.sunny, thus existence of NSSI#1, NSSI#2 and NSSI#3 with their assigned `SliceProfiles` is assumed. This time, NSSI allocation request arrives from PLMN X, who establishes an NSI covering all highways in the region, including the highway in the area covered by the considered RAN domain. The purpose of the currently built NSI is to support the trials of V2X Services conducted by automotive companies that are clients of PLMN X. The "X-Highways-V2X" `SliceProfile` sent in the `allocateNssi` request contains the following `SlsAttributes`, which are successfully validated against the `DesignPolicies` defined for PLMN X:

```
(1) Received allocateNssi request with the following SliceProfile:
{
  "serviceName" : "X-Highways-V2X",
  "pLMNInfoList" : [ "X:4" ]
  "coverageAreaTAList" : [ "TA3", "TA2", "TA4" ],
  "resourceSharingLevel" : "NON_SHARED",
  "dlLatency" : 10,
  "ulLatency" : 10,
  "dlThptPerSliceSubnet" : 500,
  "ulThptPerSliceSubnet" : 300,
  "maxNumberofUEs" : 10000,
}
[...]
(4) Validation of allocation request against DesignPolicies for PLMN 'X': SUCCESSFUL
```

It is important to note that due to the mission-critical characteristics of the V2X traffic to be supported by the allocated NSSI, the `resourceSharingLevel` *Requirement* is set to `NON_SHARED`. This means that the NSSI to be allocated should not be utilized by any other NSI (no other `SliceProfiles` should be assigned to such NSSI). Moreover, there are stringent *Requirements* values posed on the latency of the traffic expected not to exceed 20ms RTT. With such high demands, it is expected that the ASD workflow designs and deploys a new isolated NSSI with dedicated PRBs and high prioritization of uplink and downlink traffic.

Execution of the ASD workflow brings the expected effect, as visualized in Figure 62:

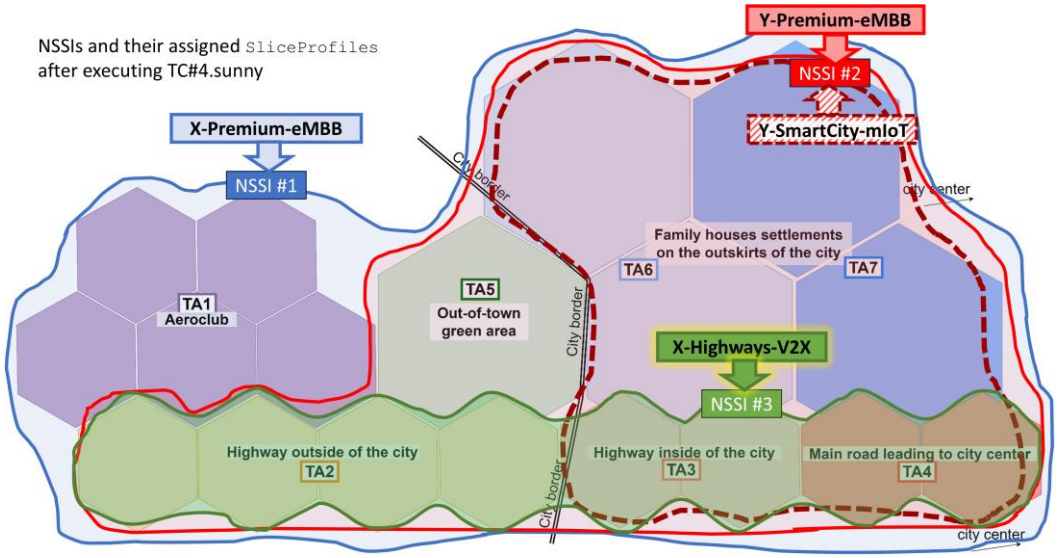


Figure 62. Geographical coverage of the provisioned NSSIs and their assigned `SliceProfiles` after executing TC#4.sunny.

In the following points, the most important aspects of ASD workflow execution are highlighted:

- In the first iteration of steps 6-16, the high-level SLS *Requirements* were decomposed by means of the `RDRs` contained in the `V2X_NSST` "translation NSST"[178], which was the only *candidate* that met the posed SLS *Requirements*. The resulting lower-level SLS *Requirements* were as follows:

```
(15) Decomposing SLS Requirements according to RDRs:
        ['Subnet'].dLLatency = 10
        ['Subnet'].uLLatency = 10
        ['Subnet'].dLThptPerSliceSubnet = 500
        ['Subnet'].maxNumberofUEs = 10000
        ['Subnet'].uLThptPerSliceSubnet = 300

        -> Decomposed SLS Requirements:
                ['Subnet'].prbPartitionMinRatioDl = 20
                ['Subnet'].prbPartitionMaxRatioDl = 80
                ['Subnet'].schedulingPriorityUl = 1
                ['Subnet'].schedulingPriorityDl = 1
```

- With the recursive execution of step 6, all the *candidates* representing a Subnet are being fetched and evaluated in step 7, including NSSI#1 and NSSI#2. However, existing NSSIs are evaluated as not fulfilling the *Requirements*, due to lack of available PRBs and the fact that they already have `SliceProfiles` assigned:

```
(6) Fetched candidates representing 'Subnet': [NSSI='NSSI#1', NSSI='NSSI#2', NSST='LOW_LEVEL_RAN_NSST', NSST='MIOT_NSST',
NSST='V2X_NSST', NSST='EMBB_NSST']
(7) Evaluation of candidates for fulfillment of SLS Requirements:
    pLMNInfoList: '[X:4]'
    resourceSharingLevel: 'NON_SHARED'
    prbPartitionMinRatioDl: '20'
    schedulingPriorityUl: '1'
    prbPartitionMaxRatioDl: '80'
    coverageAreaTAList: '[X:TA4, X:TA3, X:TA2]'
    schedulingPriorityDl: '1'

    -> Candidates that cross the evaluationThreshold = 'FULFILLS': ["LOW_LEVEL_RAN_NSST"]
        [...]
    -> Candidates that DO NOT cross the evaluationThreshold = 'FULFILLS': ["EMBB_NSST", "MIOT_NSST", "NSSI#2", "NSSI#1",
"V2X_NSST"]
    - 'NSSI#2' evaluated as 'DOES_NOT_CONTRIBUTE' with Capabilities evaluation results:
       prbPartitionMinRatioDl = '0' -> 'DOES_NOT_CONTRIBUTE'
       resourceSharingLevel = 'SHARED' -> 'DOES_NOT_CONTRIBUTE'
        [...]
    - 'NSSI#1' evaluated as 'DOES_NOT_CONTRIBUTE' with Capabilities evaluation results:
       prbPartitionMinRatioDl = '0' -> 'DOES_NOT_CONTRIBUTE'
       resourceSharingLevel = 'SHARED' -> 'DOES_NOT_CONTRIBUTE'
        [...]
(13) Selected candidate: NSST='LOW_LEVEL_RAN_NSST'
```

[178] See Table 9 in Subsection 5.6.3 for the specification of RDRs for the V2X-typed Subnets.

- New NSSI has been designed based on the `LOW_LEVEL_RAN_NSST` in a similar way as in the previously described Scenarios. However, only the MFIs comprising gNB installations #B and #C were evaluated and selected as internals of the newly designed NSSI, as they `CONTRIBUTE` to fulfillment of the `coverageAreaTAList` *Requirement* and they offered the requested V2X type of traffic in the value of the `pLMNInfoList` *Capability*:

```
(19) Evaluation of candidates for fulfillment of SLS Requirements:
    ['CUUP']:
      -> Candidates that cross the evaluationThreshold = 'CONTRIBUTES': ["CU_UP_B", "CU_UP_C"]
      - 'CU_UP_C' evaluated as 'CONTRIBUTES' with Capabilities evaluation results:
         pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]' -> 'EXCEEDS'
         coverageAreaTAList = '[Y:TA3, Y:TA4, X:TA4, X:TA3]' -> 'CONTRIBUTES'
      - 'CU_UP_B' evaluated as 'CONTRIBUTES' with Capabilities evaluation results:
         pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]' -> 'EXCEEDS'
         coverageAreaTAList = '[Y:TA2, X:TA2]' -> 'CONTRIBUTES'
      -> Candidates that DO NOT cross the evaluationThreshold = 'CONTRIBUTES': ["CU_UP_D", "CU_UP_A"]
       [...]
    ['CUCP']:
      -> Candidates that cross the evaluationThreshold = 'CONTRIBUTES': ["CU_CP_B", "CU_CP_C"]
      - 'CU_CP_B' evaluated as 'CONTRIBUTES' with Capabilities evaluation results:
         pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]' -> 'EXCEEDS'
         coverageAreaTAList = '[Y:TA2, X:TA2]' -> 'CONTRIBUTES'
      - 'CU_CP_C' evaluated as 'CONTRIBUTES' with Capabilities evaluation results:
         pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]' -> 'EXCEEDS'
         coverageAreaTAList = '[Y:TA3, Y:TA4, X:TA4, X:TA3]' -> 'CONTRIBUTES'
      -> Candidates that DO NOT cross the evaluationThreshold = 'CONTRIBUTES': ["CU_CP_A", "CU_CP_D"]
       [...]
    ['DU']:
      -> Candidates that cross the evaluationThreshold = 'CONTRIBUTES': ["DU_C2", "DU_B1", "DU_B2", "DU_C3", "DU_C4", "DU_B3",
"DU_B4", "DU_C1"]
      - 'DU_B2' evaluated as 'CONTRIBUTES' with Capabilities evaluation results:
         pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]' -> 'EXCEEDS'
         coverageAreaTAList = '[Y:TA2, X:TA2]' -> 'CONTRIBUTES'
         schedulingPriorityUl = '[1-15]' -> 'EXCEEDS'
         schedulingPriorityDl = '[1-15]' -> 'EXCEEDS'
         prbPartitionMinRatioDl = '75' -> 'EXCEEDS'
         prbPartitionMaxRatioDl = '100' -> 'EXCEEDS'
       [...]
      -> Candidates that DO NOT cross the evaluationThreshold = 'CONTRIBUTES': ["DU_D3", "DU_A1", "DU_D4", "DU_A2", "DU_D5",
"DU_A3", "DU_A4", "DU_A5", "DU_D1", "DU_D2"]
```

- After successful execution of the remaining steps of the workflow, the newly designed Subnet was deployed and provisioned with the `SliceProfile`.

After executing TC4#sunny, the state of operational objects is as indicated in Table 18:

Table 18. Case Study operational objects' tracking table – state after TC#4.sunny execution.

State of operational objects after execution of TC#4.sunny			MFI *Slicing Entities*	gNB #A		gNB #B		gNB #C			gNB #D			
				DU#A.1-5	CU-CP&UP	DU#B.1-4	CU-CP&UP	DU#C.1-2	DU#C.3-4	CU-CP&UP	DU#D.1	DU#D.2-3	DU#D.4-5	CU-CP&UP
		Current *Capabilities*:	prbPartitionMinRatioDl	85		75-20=55		75-20=55			75			
		Aggregated Requirements:	prbPartitionMinRatioDl	15		25+20=45		25+20=45			25			
			prbPartitionMaxRatio	80		max(80;80)=80		max(80;80)=80			80			
			schedulingPriorityDl	[4]		[4,8]+[1]=[4,8,1]		[4,8]+[1]=[4,8,1]			[4,8]			
			schedulingPriorityUl	[7]		[7,9]+[1]=[7,9,1]		[7,9,7]+[1]=[7,9,7,1]			[7,9]	[7,9,7]		
			pLMNInfoList	[X:1]		[X:1,Y:1]+[X:4]=[X:1,Y:1,X:4]		[X:1,Y:1,Y:3]+[X:4]=[X:1,Y:1,Y:3,X:4]			[X:1, Y:1]	[X:1,Y:1, Y:3]		

NSSI Slicing Entities			pLMNInfo List	coverage AreaTAList	resource SharingLevel	prbPartition MinRatioDl	prbPartition MaxRatioDl	scheduling PriorityDl	scheduling PriorityUl	MFIs' assignment to NSSIs and SliceProfiles
NSSI #1	Aggregated *Capabilities*: (total / current)		[X:1,3; Y:1,3]	TA1-7	SHARED	15 / 0	80 / 80	[1-15]	[1-15]	NSSI #1
	Slice Profiles:	X-Premium-eMBB	[X:1]	TA1-7	SHARED	15	80	4	7	X-Premium-eMBB
NSSI #2	Aggregated *Capabilities*: (total / current)		[X:1,3; Y:1,3]	TA2-7	SHARED	10 / 0	60 / 60	[1-15]	[1-15]	NSSI #2
	Slice Profiles:	Y-Premium-eMBB	[Y:1]	TA2-7	SHARED	10	60	8	9	Y-Premium-eMBB
		Y-SmartCity-mIoT	[Y:3]	TA3-4, 6-7	SHARED	0	30	-	11	Y-SmartCity-mIoT / Y-SmartCity-mIoT
NSSI #3	Aggregated *Capabilities*: (total / current)		[X:1,3,4; Y:1,3]	TA2-4	NON_ SHARED	20 / 0	80 / 80	[1-15]	[1-15]	NSSI #3
	Slice Profiles:	Y-Highways-V2X	[X:4]	TA2-4	NON_ SHARED	20	80	1	1	X-Highways-V2X

What is worth commenting is that even if there was an existing NSSI without any `SliceProfiles` assigned and with unused 20% of PRBs, it would not be selected in the first attempt, due to the `candidatesPriority = DESIGN-TIME;RUNTIME` which is set for PLMN X. Only if it would not be possible to design a new NSSI then an existing one would be selected for reuse.

6.3.4.6. Scenario 5

In TC#5.sunny, PLMN Y establishes another NSI of mIoT type. The Service to be supported by the NSI is a nation-wide isolated network for collecting monitoring data from all main roads. In the considered RAN domain, the NSSI is requested to cover the area of the highway (TAs 2-4), which is the same area as in the previously considered TC#4.sunny. However, the requested Slice type and QoS demands are different, the requesting PLMN is different and the environmental situation changed as well. Thus, it is worth analyzing

how the ASD workflow processes such a request. The input `SliceProfile` for Service "Y-Roads-Monitoring-mIoT" is as follows:

```
(1) Received allocateNssi request with the following SliceProfile:
{
  "serviceName" : "Y-Roads-Monitoring-mIoT",
  "pLMNInfoList" : [ "Y:3" ]
  "coverageAreaTAList" : [ "TA3", "TA2", "TA4" ],
  "resourceSharingLevel" : "SHARED",
  "ulThptPerSliceSubnet" : 100,
  "maxNumberofUEs" : 30000,
}
```

Executing TC#5.sunny in a "black-box" manner results in the assignment of the "Y-Roads-Monitoring-mIoT" `SliceProfile` to the existing NSSI#2, which is schematically visualized in Figure 63:

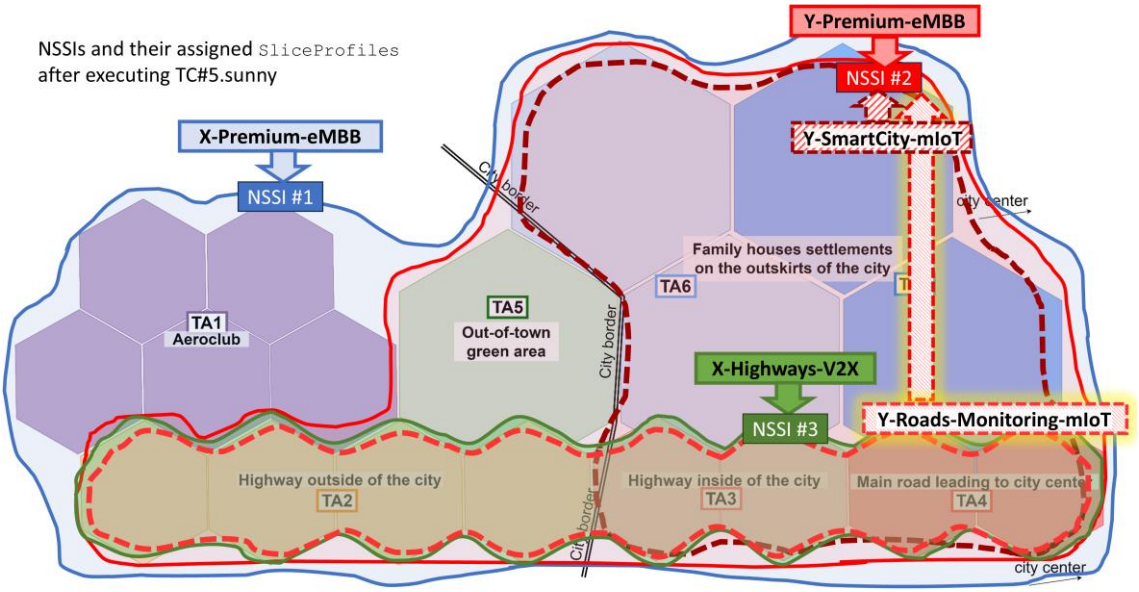


Figure 63. Geographical coverage of the provisioned NSSIs and their assigned `SliceProfiles` after executing TC#5.sunny.

In order to explain the decisions made by the ASD workflow, let's analyze the operational log:

- In the first iteration of steps 6-16, `MIOT_NSST` "translation NSST" was selected and high-level SLS *Requirements* were decomposed (similarly as in TC#2.sunny), which led to the recursive execution of step 6.

```
    [...]
(13) Selected candidate: NSST='MIOT_NSST'
    [...]
(15) Decomposing SLS Requirements according to RDRs:
       ['Subnet'].ulThptPerSliceSubnet = 100
       ['Subnet'].maxNumberofUEs = 30000
       -> Decomposed SLS Requirements:
              ['Subnet'].prbPartitionMaxRatioDl = 30
              ['Subnet'].prbPartitionMinRatioDl = 0
              ['Subnet'].schedulingPriorityUl = 11
```

- In steps 6-7, all the currently available Subnet *candidates* were fetched and evaluated. It is worth noting that NSSI#3, which matches exactly the requested coverage area and is configured with large fraction of resources (20% PRBs) that could handle the additional load, is evaluated as not crossing the `evaluationThreshold` = `FULFILLS`.

```
(6) Fetched candidates representing 'Subnet': [NSSI='NSSI#3', NSSI='NSSI#2', NSSI='NSSI#1', NSST='V2X_NSST',
NSST='MIOT_NSST', NSST='EMBB_NSST', NSST='LOW_LEVEL_RAN_NSST']
(7) Evaluation of candidates for fulfillment of SLS Requirements:
   schedulingPriorityUl = '11'
   prbPartitionMinRatioDl = '0'
   prbPartitionMaxRatioDl = '30'
   resourceSharingLevel = 'SHARED'
   pLMNInfoList = '[Y:3]'
   coverageAreaTAList = '[Y:TA2, Y:TA3, Y:TA4]'

   -> Candidates that cross the evaluationThreshold = 'FULFILLS': ["NSSI#2", "NSSI#1", "LOW_LEVEL_RAN_NSST"]
      - 'NSSI#2' evaluated as 'FULFILLS' with Capabilities evaluation results:
         pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]' -> 'EXCEEDS'
         schedulingPriorityUl = '[1-15]' -> 'EXCEEDS'
         schedulingPriorityDl = '[1-15]' -> no requirement posed
         prbPartitionMinRatioDl = '0' -> 'FULFILLS'
         resourceSharingLevel = 'SHARED' -> 'FULFILLS'
         prbPartitionMaxRatioDl = '60' -> 'EXCEEDS'
         coverageAreaTAList = '[Y:TA2, Y:TA3, Y:TA4, Y:TA5, X:TA7, X:TA6, X:TA5, X:TA4, X:TA3, X:TA2, Y:TA6, Y:TA7]' ->
'EXCEEDS'
      - 'NSSI#1' evaluated as 'FULFILLS' with Capabilities evaluation results:
         pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]' -> 'EXCEEDS'
         schedulingPriorityUl = '[1-15]' -> 'EXCEEDS'
         schedulingPriorityDl = '[1-15]' -> no requirement posed
         prbPartitionMaxRatioDl = '80' -> 'EXCEEDS'
```

```
        prbPartitionMinRatioDl = '0' -> 'FULFILLS'
        resourceSharingLevel = 'SHARED' -> 'FULFILLS'
        coverageAreaTAList = '[X:TA7, X:TA6, Y:TA6, Y:TA7, Y:TA2, Y:TA3, Y:TA4, Y:TA5, X:TA5, X:TA4, X:TA3, X:TA2, X:TA1]' -
> 'EXCEEDS'
    - 'LOW_LEVEL_RAN_NSST' evaluated as 'FULFILLS' with Capabilities evaluation results:
        schedulingPriorityUl = '[1-15]' -> 'EXCEEDS'
        prbPartitionMinRatioDl = '100' -> 'EXCEEDS'
        pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]' -> 'EXCEEDS'
        schedulingPriorityDl = '[1-15]' -> no requirement posed
        coverageAreaTAList = '[X:TA7, X:TA6, Y:TA6, Y:TA7, Y:TA2, Y:TA3, Y:TA4, Y:TA5, X:TA5, X:TA4, X:TA3, X:TA2, X:TA1]' -
> 'EXCEEDS'
        prbPartitionMaxRatioDl = '100' -> 'EXCEEDS'
        resourceSharingLevel = 'SHARED' -> 'FULFILLS'

  -> Candidates that DO NOT cross the evaluationThreshold = 'FULFILLS': ["EMBB_NSST", "MIOT_NSST", "NSSI#3", "V2X_NSST"]
    - 'NSSI#3' evaluated as 'DOES_NOT_CONTRIBUTE' with Capabilities evaluation results:
        pLMNInfoList = '[X:4, X:3, Y:1, Y:3, X:1]' -> 'EXCEEDS'
        schedulingPriorityUl = '[1-15]' -> 'EXCEEDS'
        schedulingPriorityDl = '[1-15]' -> no requirement posed
        prbPartitionMaxRatioDl = '80' -> 'EXCEEDS'
        coverageAreaTAList = '[Y:TA2, Y:TA3, Y:TA4, X:TA4, X:TA3, X:TA2]' -> 'EXCEEDS'
        prbPartitionMinRatioDl = '0' -> 'FULFILLS'
        resourceSharingLevel = 'NON_SHARED' -> 'DOES_NOT_CONTRIBUTE'
```

- NSSI#3 is evaluated as not fulfilling the *Requirements* because its `resourceSharingLevel` *Capability* is `NON_SHARED` (due to the fact that the "X-Highways-V2X" `SliceProfile` is assigned to it). This fact blocks NSSI#3 from being selected, which is the expected behavior, as NSSI#3 is dedicated for exclusive utilization for the mission-critical V2X Service.
- Due to the configured `candidatesPriority` = `RUNTIME;DESIGN-TIME`, the existing NSSIs (NSSI#1 and NSSI#2) were considered for selection in steps 8-10. For PLMN Y, maximization of existing NSSIs utilization is configured by the `runtimeCandidatesPriority` set to `HIGHEST_REQUIREMENTS_COUNT`. NSSI#1 is currently assigned with a single `SliceProfile`, while NSSI#2 already supports two, but still offers enough capacity to handle additional load. Thus, NSSI#2 got selected for the allocation of the currently considered `SliceProfile`.

The state of operational objects after execution of TC5#.sunny is summarized in Table 19:

Table 19. Case Study operational objects' tracking table – state after TC#5.sunny execution.

State of operational objects after execution of TC#5.sunny	MFI *Slicing Entities*	gNB #A		gNB #B		gNB #C			gNB #D			
		DU#A.1-5	CU-CP&UP	DU#B.1-4	CU-CP&UP	DU#C.1-2	DU#C.3-4	CU-CP&UP	DU#D.1	DU#D.2-3	DU#D.4-5	CU-CP&UP
	Current *Capabilities*: prbPartitionMinRatioDl	85		55-0=55		55-0=55			75			
	Aggregated Requirements: prbPartitionMinRatioDl	15		45+0=45		45+0=45			25			
	prbPartitionMaxRatio	80		max(80;30)=80		max(80;30)=80			80			
	schedulingPriorityDl	[4]		[4,8,1]		[4,8,1]			[4,8]			
	schedulingPriorityUl	[7]		[7,9,1]+[11]=[7,9,1,11]		[7,9,7,1]+[11]=[7,9,7,1,11]			[7,9]	[7,9,7]		
	pLMNInfoList	[X:1]		X:1,Y:1,X:4]+[Y:3]=[X:1,Y:1,X:4,Y:3		[X:1,Y:1,Y:3,X:4]+[Y:3]=[X:1,Y:1,Y:3,X:4,Y:3]			[X:1, Y:1]	[X:1,Y:1, Y:3]		

NSSI Slicing Entities			pLMNInfo List	coverage AreaTAList	resource SharingLevel	prbPartition MinRatioDl	prbPartition MaxRatioDl	scheduling PriorityDl	scheduling PriorityUl	MFIs' assignment to NSSIs and SliceProfiles
NSSI #1	Aggregated *Capabilities*: (total / current)		[X:1,3; Y:1,3]	TA1-7	SHARED	15 / 0	80 / 80	[1-15]	[1-15]	NSSI #1
	Slice Profiles:	X-Premium-eMBB	[X:1]	TA1-7	SHARED	15	80	4	7	X-Premium-eMBB
NSSI #2	Aggregated *Capabilities*: (total / current)		[X:1,3; Y:1,3]	TA2-7	SHARED	10 / 0	60 / 60	[1-15]	[1-15]	NSSI #2
	Slice Profiles:	Y-Premium-eMBB	[Y:1]	TA2-7	SHARED	10	60	8	9	Y-Premium-eMBB
		Y-SmartCity-mIoT	[Y:3]	TA3-4, 6-7	SHARED	0	30	-	11	Y-SmartCity-mIoT / Y-SmartCity-mIoT
		Y-Roads-Monitoring-mIoT	[Y:3]	TA2-4	SHARED	0	30	-	11	Y-Roads-Monitoring-mIoT
NSSI #3	Aggregated *Capabilities*: (total / current)		[X:1,3,4; Y:1,3]	TA2-4	NON_ SHARED	20 / 0	80 / 80	[1-15]	[1-15]	NSSI #3
	Slice Profiles:	Y-Highways-V2X	[X:4]	TA2-4	NON_ SHARED	20	80	1	1	X-Highways-V2X

After executing TC#5.sunny, there are 3 NSSIs deployed and 5 NSIs running on top of them, which are represented by the `SliceProfiles` assigned to the NSSIs.

6.3.5. Functional evaluation conclusions

The Case Study presented in this Section gave a comprehensive overview of the ASD workflow's operation, contributing towards consolidated understanding of its working mechanisms. The executed TCs exercised the proper functioning of the automated ASD workflow, by exposing it to a representative set of conditions in an operational environment resembling a potential, real Network Slicing deployment in a sample RAN domain. Among multiple insights revealed by the conducted study, the gathered results allow stating several conclusions that are **important for verifying the thesis statements** formulated at the beginning of this dissertation. The fulfillment of functional requirements the sought solution should meet, as stated in Section 2.4, can be assessed as well.

The step-by-step analysis of the workflow's functioning and the effects brought in the state of the orchestrated *Slicing Entities* allowed for verifying the design decisions undertaken by the individual activities and their entire flow, **confirming that the results produced in the particular circumstances are valid**. It is claimed so, because the ASD workflow correctly decides on the assignment of the incoming `SliceProfiles` either to newly designed NSSIs or to the already existing ones, by taking into account the demanded characteristics of the NSSI to be allocated, the *Capabilities* of the currently available *Slicing Entities* and the configuration of the `DesignPolicies`, as justified in the paragraphs below.

The execution of the "rainy-day" TCs allowed for verifying the ASD workflow's resiliency to invalid inputs and correct refusal of the allocation requests that in the particular environmental conditions couldn't be handled. In turn, all the considered "sunny-day" TCs, the ASD workflow produced results that were accepted as valid input for the "New NSSI provisioning" and "NSSI re-configuration for `SliceProfile` allocation" procedures in phases D and E – this fact serves as an ultimate determinant of design **results correctness**. Here it needs to be commented that there is not only a single design result that can be perceived correct in the given conditions – there may be multiple variants of correct design results. It is because the design process is influenced by the established `DesignPolicies`, which may prioritize different aspects of the Customer's, Operator's and Provider's operational&business objectives. For example, although a decision of reusing an already existing, loaded NSSI in the particular conditions may not be the best in terms of the offered QoS, it will be satisfactory from the Operator's and the Customer's perspectives since the exploitation of RAN resources is maximized and the Customer's cost of allocating an NSSI is minimized.

Another observation is that the implemented solution **relieves the Operator from a number of tedious activities, analysis and decision-making efforts that require focus and expert knowledge**. Inspection of all the available *Slicing Entities*, their current operational state and offered *Capabilities* is complex and would be very laborious to be performed by a human Operator even in small scale. This becomes even more apparent in a larger scale of operated environments, where the current state of *Capabilities* of hundreds of *Slicing Entities* and their currently assigned `RequirementsSets` need to be analyzed. The automatically performed evaluation, selection, decomposition and aggregation of candidates' *Capabilities* relieves the human Operator from performing these duties, which with the growing scale of administered domains and increasing number of *Slicing Entities* would be unmanageably complex and, as a result, infeasible. Thanks to automation, all these complex, error-prone procedures are concealed and human Operator's effort is saved.

All the observations mentioned above allow for concluding that the concepts of the elaborated automation framework **enabled the construction of an automation solution that delivers the assumed functionality** and **produces the expected results**. As shown in the Case Study, once the constructed solution is configured, it works autonomously, **excluding the need for human Operator participation** in the design process and, consequently, in the entire end-to-end allocation process. Thus, it has been shown that the embodiment of the proposed framework **indeed enables** automated transition from high-level, user-oriented SLS *Requirements* to the technical specifications of NSSIs to be allocated, which **fulfills the promises stated in the thesis** and meets all the functional requirements against the design automation solution stated in Section 2.4.

At the same time, author humbly admits that the described Case Study exposed the ASD workflow only to a tiny fraction of operational circumstances in which it may need to operate when deployed in various industrial conditions. It is because the space of variables in the Customer's requests (i.e. `SliceProfile` attributes and their values) and in the Subject Orchestrator's operational state (i.e. state of all operational objects, configuration of `DesignPolicies`, and all the models conveying domain knowledge) is huge. The performance evaluation TCs described in the next Section reinforce the functional verification by exposing the PoC solution to a larger number of scenarios in different operational conditions, thanks to which the correct operation of the ASD PoC is confirmed for a larger set of data. From this perspective, the author claims that the breadth of the conducted tests is sufficient to **confirm correct functioning of the elaborated concepts** applied in the context of the Subject Orchestrator and in the environmental conditions that are anticipated to be supported in its operational functioning.

6.4. Part 2 - performance evaluation

In order to further strengthen the verification of the functional aspect and to provide an insight into selected non-functional features of the developed PoC solution, its further empirical examination based on a broader spectrum of input data is justified. Hence, in this Section, a combination of functional and performance evaluation is described. Firstly, several remarks and assumptions are stated and the focus of the conducted tests is clarified. Next, the synthetic input data utilized in the experiments is introduced, based on which the Scenarios and Test Cases are presented. Then, selected details of tests execution are provided and the obtained results are discussed. Lastly, conclusions from the conducted experiments are stated.

6.4.1. Clarification of experiments' scope and assumptions

The evaluation described in this Section investigates **how the performance of the design process is impacted by the growing scale of NSSIs being handled** and verifying that the produced results are valid. In the experiments, the scale of NSSI(s) is considered from two perspectives: (1) the scale of an NSSI-**to**-**be**-**allocated** and (2) the scale of NSSIs **already allocated** at the moment of receiving the considered allocation request from the Customer. However, it is important to explain what is meant by "scale of NSSI(s)" in these two perspectives and why the adopted perceptions are indicative for the evaluation of the design process performance.

The (1) "scale of an NSSI-to-be-allocated" is measured by the count and organization of *Slicing Entities* out of which it is designed. Such measure is adopted as it reflects the quantity and quality of resources that need to be analyzed and processed during the design process to allocate an NSSI. In turn, the (2) "scale of NSSIs already allocated" is measured by the count and organization of operational objects in the Subject Orchestrator's inventories related to the management of the already deployed NSSIs, presence of which at scale impacts the performance of the design process. More specifically, these are the count of currently utilized *Runtime Slicing Entities*, count and semantics of `SlsAttributes` exposed in their `CapabilitiesSets` as well as count of `RequirementsSets` (`SliceProfiles` and their decomposed, lower-level counterparts) assigned to them[179]. Collectively, all these factors can be referred to as the Subject Orchestrator's **operational load** experienced at the moment of receiving the considered allocation request. The evaluation focuses on **identifying the factors** of operational load that are the most influential on the performance of the design process and finding out the ASD workflow's **sensitivity** to their scaling.

Here it is important to underline that the focus of the performance evaluation is on the implemented PoC elements, not on the Kubernetes platform the solution runs on and any tuning that could be achieved with its tooling. As such, the performance testing does not aim at investigating how the pods of the Subject Orchestrator's components scale up/down or in/out when crossing some operational thresholds. Conversely, the tests aim at **revealing the operational limits** that can be handled by the ASD workflow in an installation of the Subject Orchestrator consisting of single pods of the RAN NSSMF, SLSM and NFMF components. The discovered limits may be utilized as thresholds for performing further tests involving scaling the pods of the particular components or scaling in/out the entire Subject Orchestrator's installation. However, such tests are out of scope of this thesis and, as such, are considered as a future work.

In this context, the operational load possible to be handled by the Subject Orchestrator needs to be commented as well. According to its operational dimensioning, a single installation of the Subject Orchestrator is capable of handling 256 MFIs with 768 MLIs between them, based on which 256 NSSIs can be deployed, each maximally assigned with 6 `SliceProfiles`. This gives a theoretical barrier of handling 1536 NSIs simultaneously[180]. Although crossing these limits is possible, the Subject Orchestrator may become non-responsive and crash. Thus, the aforementioned limits are taken as operational boundaries when establishing the operational conditions in the performance evaluation.

Since the focus of this thesis and the PoC is on the design activities executed in Phases A, B and C, the provisioning procedures executed in phases (D) and (E) are **reused as implemented** originally in the Subject Orchestrator. Although these procedures are internally complex, in the experiments they are evaluated from

[179] See Subsection 5.5.2.4 for an illustration of operational objects count and arrangement in an example, simple scenario.

[180] The count of `SliceProfiles` registered in the system represents the count of NSIs currently allocated to the provisioned NSSIs.

a black-box perspective by only measuring the total time spent on their execution, without more granular investigation. Moreover, the results of the conducted tests show that although functionally valid, implementation of these procedures in terms of performance is suboptimal, which results in long times of processing in the case of large-scale NSSI deployments (details provided in the upcoming Subsection 6.4.5). However, due to the fact that this dissertation's merit is about the design process, tweaking or tuning the implementation of the procedures executed in Phases D and E is out of scope of the experiments.

Lastly, it is important to mention that database caches utilization is assumed in the experiments, in order to focus the time measurements on data processing, rather than its obtaining.

6.4.2. Conditions of the experiments

The organization of the performance experiments revolves around the combination of two aspects:

(1) the `SliceProfile` provided in the `allocateNssi` invocation by the Customer, where the value of the `coverageAreaTAList` *Requirement* determines the scale of an NSSI-to-be-allocated
(2) the arrangement and configuration of resources in the administered RAN domain and the operational load experienced by the Subject Orchestrator which represents the scale of NSSIs already allocated

In order to prepare indicative Scenarios concerning these two aspects, the Subject Orchestrator is fed with multiple sets of input data, as schematically depicted in Figure 64.

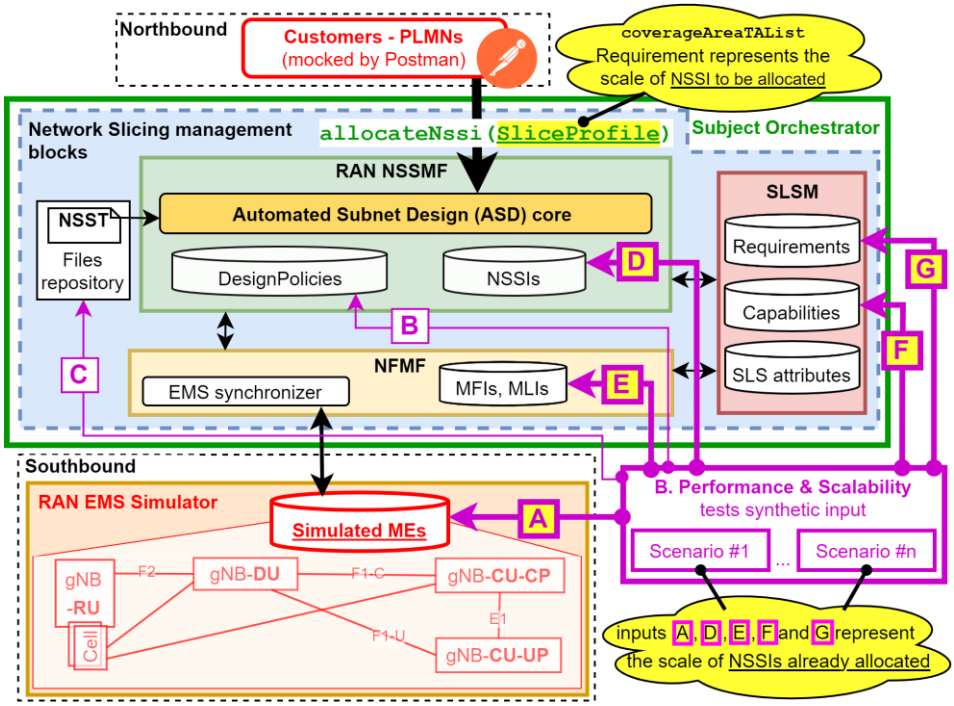


Figure 64. Inputs feeding the experimental testbed for the performance evaluation.

Inputs B (`DesignPolicies`) and C (`NSSTs`) are the same as utilized in the Case Study (see Subsection 6.3.2.2 and 6.3.2.3). The rest of the inputs (A, D-G) are described in the following Subsections.

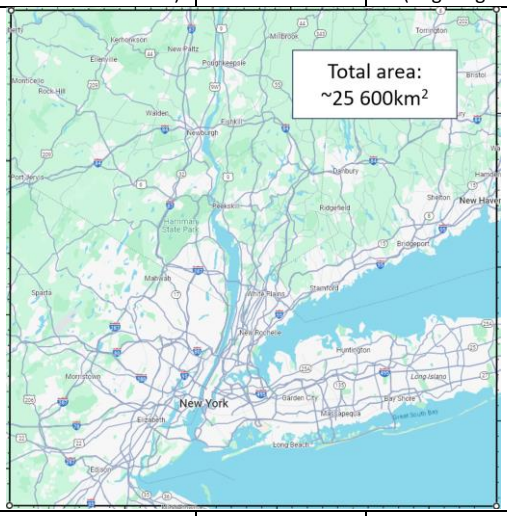
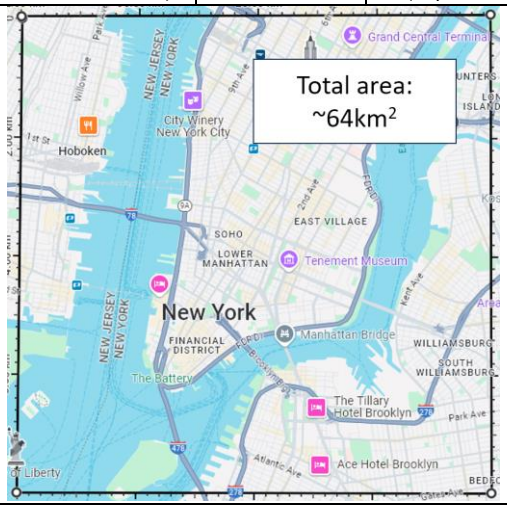
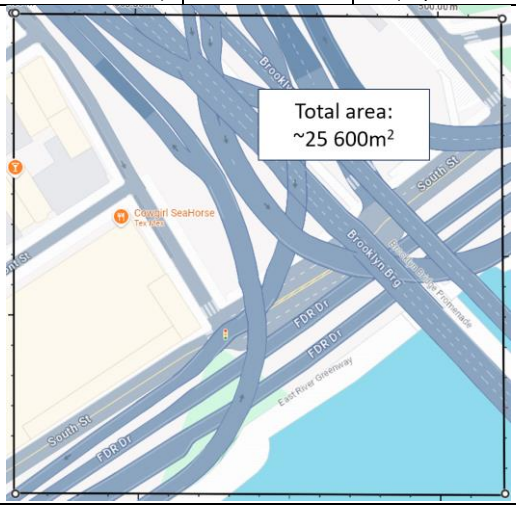
6.4.2.1. Control-plane of the RAN infrastructure – gNB NEs synthetic data

Input A delivers the ME objects that represent the control-plane of the gNB installations comprising the infrastructure of a **large-scale RAN domain**. For this purpose, EMS Simulator is fed with data mimicking 8 gNB installations[167], each consisting of 64 DUs connected to single instances of CU-CP and CU-UP. It is assumed that each DU is connected to a single Radio Unit (RU) serving a single Cell that belongs to a single Tracking Area (TA)[61]. Thanks to such an assumption, the number of TAs passed by the Customer in the `SliceProfile.coverageAreaTAList` attribute directly corresponds to the number of DUs and CU-UPs/CU-CPs of an NSSI to be allocated. Hence, the scale of NSSI-to-be-allocated in the tests can be easily determined by manipulating the value of this single SLS *Requirement*.

Although the total number of utilized MEs (528) exceeds the Subject Orchestrator's operational limit mentioned in Subsection 6.4.1, in the experiments the MFI *Slicing Entities* created based of these MEs are divided into two pools: 264 **loaded** MFIs (gNB installations #1-4) and 264 **not loaded** MFIs (gNB installations #5-8). The loaded MFIs represent the internals of the already existing NSSI deployments that are pre-assigned with a given count of `SliceProfiles` [181]. In turn, the not loaded MFIs are not assigned with any `RequirementsSets`, and therefore do not contribute to the Subject Orchestrator's load. Such arrangement of the MFIs' pools allows for checking the ASD workflow's performance in scenarios where, in the given conditions, the NSSI-to-be-allocated is designed from (if new NSSI is created) or consists of (if existing NSSI is reused) *Slicing Entities* that are already assigned with various count of `RequirementsSets`.

It shall be noted that the gNB MEs synthetic input data utilized in the experiments has been organized purely for the purpose of the performance tests and does not necessarily resemble the organization of resources in a real RAN domain. In the real RAN deployments, the arrangement and sizes of Cells and their assignments into TAs depend on multiple operational factors that can significantly differ between the particular Providers and considered domains (e.g. reception characteristics of the terrain covered by the RU, planned network capacity, intended connectivity quality, operational policies, etc.). Nevertheless, to illustrate the potential scale of the described RAN domain, the juxtaposition presented in Table 20 provides an estimation of its potential geographical coverage in various configurations, assuming 256 uniformly distributed Cells are managed.

Table 20. Estimation of RAN domain geographical coverage assuming uniform distribution of 256 Cells in various configurations.

Cells' average diameter	**Frequency bands**	**Aproximate coverage area**	**Cells' average diameter**	**Frequency bands**	**Aproximate coverage area**
10 km (macro-cellular network)	700-800 MHz	~25 600 km² (large region)	2 km (micro-cellular network)	2,5 GHz	~1024 km² (large agglomeration)
					
500 m (pico-cellular network)	3,5-5 GHz	64 km² (city district)	10 m (femto-cellular network)	24-40 GHz	25 600 m² (city settlement)
					

[181] In other words, `CapabilitiesSets` of these NSSIs and their internal MFIs are assigned with a predefined count of `RequirementsSets`.

The synthetic gNB MEs data described above allows for considering a significantly complex organization of *Slicing Entities* inside of a single orchestrated RAN domain and, as such, can serve as a basis for mimicking preexisting NSSI deployments of various scales.

6.4.2.2. Subject Orchestrator's load conditions - scale of NSSIs already allocated

Based on the RAN infrastructure data described in the previous Subsection, the Subject Orchestrator's inventories can be fed with various combinations of operational objects, represented in Figure 64 as inputs D (NSSIs), E (MFIs and MLIs), F (`CapabilitiesSets`) and G (`RequirementsSets`). The arrangement and count of these operational objects may represent various scales of brownfield Network Slicing deployments in the considered RAN domain, representing various operational load experienced by the Subject Orchestrator at a given moment. Among a range of indicators of such load, various quantitative and qualitative measures can be considered, with several examples presented in Table 21:

Table 21. Examples of Subject Orchestrator's operational load indicators considered in the performance evaluation.

Quantitative (**δ**)	Qualitative (**ε**)[182]
• δ_1: count of operated NSSIs • δ_2: count of **`SliceProfiles`** assigned to each NSSI[180] • δ_3: count of operated *lowest-level Slicing Entities* (MFIs, MLIs) • δ_4: total count of **`RequirementsSets`** (**`SliceProfiles`** + **`RequirementsSets`** assigned to *lowest-level Slicing Entities*) • δ_5: count of supported **`SlsDefinitions`**	• ε_1: distribution of **`SliceProfiles`** over the existing NSSIs • ε_2: distribution of **`RequirementsSets`** over the *lowest-level Slicing Entities* • ε_3: supported types of *lowest-level Slicing Entities* • ε_4: semantics and way of operational handling of supported **`SlsDefinitions`**

Capturing all the possible variants of such indicators is infeasible, as the spectrum of combinations of their values is tremendous. To reduce the considered space in the conducted tests, values of some indicators are considered constant (marked in Table 21 with blue font) and a particular arrangement of the qualitative indicators is assumed (purple font). More specifically, indicators δ_3 and ε_3 are taken as constant, because all of the established load conditions are based on the same data representing gNB installations of the considered RAN domain (input A) described in Subsection 6.4.2.1. The count of supported `SlsDefinitions` (δ_5), their semantics and way operational handling (ε_4) are also taken as constant, as they correspond to the already described characteristics of the SLS *attributes* supported in the PoC.

To further narrow down the considered spectrum of variables, the arrangement of values for some of the indicators is assumed to be dependent on the value of other indicators. More specifically, the distribution of `SliceProfiles` over the NSSIs (ε_1) is assumed to be uniform, which means that in the given load conditions each NSSI is assigned with the same number of `SliceProfiles`. This, in turn, induces uniform distribution of `RequirementsSets` over their internal MFIs (ε_2), making the total count of `RequirementsSets` managed by the Subject Orchestrator (δ_4) directly dependent on indicators δ_1 (count of NSSIs) and δ_2 (count of `SliceProfiles` assigned to the each NSSI). Thanks to the described assumptions, the spectrum of indicators selected for manipulation in the experiments has been limited to two quantitative measures (green font): the count of operated NSSIs (δ_1) and the count of `SliceProfiles` assigned to each NSSI (δ_2). Such an approach allows for establishing a sound range of load conditions that exercise the developed PoC solution against various scales of the NSSIs-already-allocated.

Table 22. Specification of load conditions of the Subject Orchestrator.

load conditions ID (**α**nssi_**β**sp)=**ω**rs	(**α**) count of NSSIs	(**β**) count of `SliceProfiles` assigned to each NSSI	(**ω**) total count of `RequirementsSets`
(64nssi_0sp)=0rs	64	0	0
(64nssi_2sp)=512rs	64	2	512
(64nssi_4sp)=1024rs	64	4	1024
(64nssi_6sp)=1536rs	64	6	1536
(128nssi_0sp)=0rs	128	0	0
(128nssi_2sp)=1024rs	128	2	1024
(128nssi_4sp)=2048rs	128	4	2048
(128nssi_6sp)=3072rs	128	6	3072
(256nssi_0sp)=0rs	256	0	0
(256nssi_2sp)=2048rs	256	2	2048
(256nssi_4sp)=4096rs	256	4	4096
(256nssi_6sp)=6144rs	256	6	6144

The author established 12 combinations of the mentioned indicators, as specified in Table 22. A given load condition is identified by the following pattern: (**α**nssi_**β**sp)=**ω**rs, where **α** is

[182] It is worth noting that the values of the particular `SlsAttributes` (representing both *Requirements* or *Capabilities*) are not considered among the qualitative load indicators. Only the count of the `SlsDefinitions` (δ_5) is indicative as the scale of Subject Orchestrator's load because it imposes performing correspondingly more or less computations while processing the `CapabilitiesSets`/`RequirementsSet` that contain the `SlsAttributes`. In turn, the value of a particular `SlsAttribute` (*Capability* or *Requirement*) pertains to the operational configuration of a given *Slicing Entity*, which potentially may have an impact on its operational load, but not on the Subject Orchestrator's.

the value of indicator δ_1 (count of NSSIs), **β** is the value of indicator δ_2 (count of `SliceProfiles` (sp) assigned to each NSSI) and **ω** is the value of indicator A4 (total count of `RequirementsSets` (rs) managed by the Subject Orchestrator). The "=" sign is utilized, because the ω depends on α and β. All the load conditions have been firstly generated in a synthetic manner and then stored as database dumps of the RAN NSSMF, NFMF and SLSM components. Hence, inputs D-G depicted in Figure 64 represent database dumps that allow feeding the Subject Orchestrator's inventories for repetitive execution of various Test Cases in identical load conditions.

Determination of the concretized counts of the enumerated variables by which the already allocated NSSIs can be perceived of small- or large-scale is not strict and will depend on the environmental conditions of the considered RAN domain, features of the utilized *Slicing Entities* and the Operator's operational policies. In this thesis, the scale of the already allocated NSSIs is perceived as large when the count of operated NSSIs, utilized MFIs and assigned `RequirementsSets` is close to the operational limits of the Subject Orchestrator's dimensioning mentioned in Section 6.4.1.

6.4.3. Test Cases specification

The infrastructure of the RAN domain and the Subject Orchestrator's load conditions described in the previous Subsections serve as a basis for specifying the Test Cases (TC) to be executed. As a primary focus of the conducted experiments, the ASD workflow performance is exercised against the varying scale of NSSI-to-be-allocated. However, it must be reminded that the design decisions regarding the allocated NSSI may differ depending on the circumstances in which the request is being processed, which is influenced by the `DesignPolicies`, current operational load, etc. Thus, the established TCs also take the following two dimensions into account: the fact if the **NSSI is newly designed** or an **existing NSSI is reused**, as well as the fact if the allocated NSSI and its internal MFIs are already **loaded** with `RequirementsSets` or **not**[183]. By considering the 3 dimensions mentioned above, 20 TCs labeled from A to T have been established as schematically depicted in Figure 65 and commented in the paragraphs below.

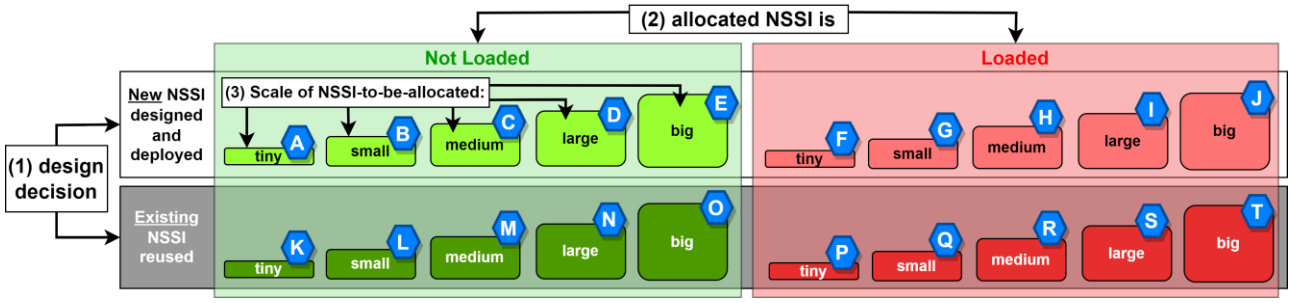


Figure 65. Classification of performance evaluation Test Cases.

According to the first dimension, the TCs can be divided into those where the design decision is to allocate a **new** or an **existing** NSSI. In TCs A-J, the input `SliceProfile` is prepared in such a way, that in the given load conditions there is no existing NSSI *candidate* capable of fulfilling them. Thus, the design process is expected to produce an `InstanceDescriptor` of a new NSSI and pass it to the procedures of phases D and E that follow in the allocation workflow. In turn, in TCs K-T, the input `SliceProfile` is constructed in such a way that the design process will point to reuse an existing NSSI that is capable of fulfilling it.

Secondly, the TCs can be categorized into those resulting in the allocation of a **loaded** or a **not loaded** NSSI. More specifically, in TCs A-E the new NSSI will be designed from MFIs belonging to the not loaded pool. In scenarios K-O, the existing NSSI selected to be reused is not loaded with any `RequirementsSets`. In contrast, in TCs F-J, a new NSSI will be designed from MFIs that are already loaded with a given count of `RequirementsSets` (according to the considered load conditions), but still offer satisfactory *Capabilities* values to accommodate the input `SliceProfile`. In turn, in TCs P-T, the existing NSSI selected to be reused is already loaded with other `SliceProfiles` attached to it before, but exposes *Aggregated Capabilities* values that are sufficient to satisfy the input `SliceProfile`.

[183] In the performance evaluation TCs, NSSI's and MFI's load is considered in a binary manner – loaded or not loaded. This indicates if the NSSI is designed/already built from the loaded or not loaded pool of MFIs. How heavily the NSSIs and MFIs are loaded with `RequirementsSets` is determined by the operational load conditions specified in Subsection 6.4.2.2.

Finally, the TCs are differentiated according to the **scale of NSSI-to-be-allocated**, in terms of the count of its internal MFIs. Determination of the concretized count of MFIs by which an NSSI-to-be-allocated should be perceived small, mid or large-scale is not strict and may differ depending on the considered RAN domain, Operator's policies and other factors. However, for the sake of the conducted experiments in the considered conditions, the assumed scales of NSSI-to-be-allocated are listed in Table 23.

Table 23. Specification of scale of NSSI-to-be-allocated in the performance evaluation Test Cases.

NSSI **scale**	**Count** and arrangement of MFIs	Approx. **coverage area** of the RAN domain
tiny	**4** MFIs: 2 DUs, 1 CU-CP, 1 CU-UP	1%
small	**12** MFIs: 10 DUs, 1 CU-CP, 1 CU-UP	5%
medium	**66** MFIs: 64 DUs, 1 CU-CP, 1 CU-UP	25%
large	**132** MFIs: 128 DUs, 2 CU-CPs, 2 CU-UPs	50%
big	**264** MFIs: 256 DUs, 4 CU-CPs, 4 CU-UPs	100%

It is important to note that although the TCs specifications focus on manipulating the metrics related to performance, by conducting such tests, the developed PoC solution is also exposed to functional scenarios that are different from the Case Study and the TCs are executed in different environmental conditions. Therefore, the experiments discussed in this Section are **also treated as an extended functional evaluation** that further contributes to verifying the expected functioning of the implemented concepts on a broader set of input data.

6.4.4. Selected details of Test Cases execution

Similarly as in the Case Study, also in this part of the evaluation each TCs is about invoking the `allocateNssi(SliceProfile)` operation with different values of the input SLS *Requirements*. However, unlike in the Case Study, where several "rainy-day" TCs have been considered, here all the allocation requests and sets of input data have been synthetically crafted to succeed, inducing the desired flow of the design process according to the specification of a given TC[184]. For example, in TC-A, the `SliceProfile` is as follows:

```
{
   "serviceName" : "TC-A",
   "coverageAreaTAList" : [ "TA1", "TA2" ],
   "resourceSharingLevel" : "SHARED",
   "dlThptPerSliceSubnet" : 300,
   "maxNumberofUEs" : 100000,
   "pLMNInfoList" : [ "Y:1" ]
}
```

After initial *Requirements* decomposition based on the eMBB "translation NSST", the following set of lower-level SLS *Requirements* is considered in the design process:

```
(15) Decomposing SLS Requirements according to RDRs:
        ['Subnet'].dlThptPerSliceSubnet = 300
        ['Subnet'].maxNumberofUEs = 100000

        -> Decomposed SLS Requirements:
                ['Subnet'].prbPartitionMaxRatioDl = 80
                ['Subnet'].prbPartitionMinRatioDl = 5
                ['Subnet'].schedulingPriorityDl = 12
```

The `SliceProfiles` for other TCs are similar – an eMBB-typed NSSI with the mentioned QoS characteristics is requested to be allocated by PLMN Y which results in the consumption of 5% of the available PRBs in the given location. Only the value of the `coverageAreaTAList` is manipulated in the particular TCs in order to achieve the allocation of an NSSI of a desired scale. In the TCs where the loaded or not loaded MFIs are utilized, their corresponding TAs are provided as the `coverageAreaTAList` value. Moreover, the Subject Orchestrator's load conditions (inputs D-G in Figure 64) have been prepared in such a way that the execution of the particular allocation request in a given TC results in the allocation of an existing or new NSSI as specified in a particular TC.

The `DesignPolicies` utilized for the execution of all TCs are the ones defined for PLMN Y (see Subsection 6.3.2.2), which means that the existing NSSI candidates are firstly attempted to be allocated and only when none of them meets the input SLS *Requirements*, a new one is designed. Such approach allows for investigating the ASD workflow in its entire extent, measuring the total time devoted to fetching and evaluation of existing NSSI *candidates* revealed in the execution of TCs A-J. In contrast, in TCs K-S, which result in the allocation of an existing NSSI, the NSSIs evaluation is interrupted once a *candidate* satisfying the selection criteria is found[185], which can occur at various moments while analyzing the *candidates*.

[184] No "rainy-day" TCs are considered as unsuccessful design cases are not indicative for performance and scalability evaluation.

[185] Measurements for *evaluation* and *selection* are combined due to the feedback mechanism implemented in the PoC (see Subsection 5.5.3.7).

For measuring the processing times, the code of the ASD PoC has been equipped with a number of measuring probes placed around the particular design activities as well as the provisioning and allocation procedures in an Aspect-oriented manner. Due to the fact that the provisioning and allocation procedures in phases (D) and (E) are not the main areas of interest of this thesis and in the experimentations, they are treated from a coarse grained, "black box" perspective, without placing more granular time measuring probes inside of their logic. Nevertheless, capturing those times allows investigating the proportions of time spent in the design activities, compared to the time consumed by the provisioning and allocation procedures that follow in the end-to-end allocation workflow.

In order to ensure a credible assessment of the solution's performance, each of the 20 TC has been executed 5 times in each of the 12 Subject Orchestrator's load conditions defined in Subsection 6.4.2.2. Out of the 5 measurements, the outliers have been rejected and an average has been calculated out of the remaining 3 measurements. Such an approach allowed for ignoring any accidental fluctuations in the runtime platform's performance and it also allowed for taking into account the influence of database caching utilized in the majority of the design activities. As such, the tests are better focused on measuring the processing of the data rather than its fetching, minimizing the Input-Output (IO) delays that constitute the usual bottlenecks in any computer system.

6.4.5. Obtained results and their analysis

In this Section, the results obtained from the executed experiments are presented and discussed. While there are several perspectives from which the gathered results can be analyzed (e.g. by Test Case, by a chosen dependent variable, etc.), presenting each of such perspective on a separated set of charts would occupy a significant volume of this dissertation. Thus, for the sake of brevity and clarity, a single perspective is adopted - each chart presents the times of executing a given TC in all of the established Subject Orchestrator's load conditions. Nevertheless, in the charts, several "areas of interest" are distinguished in order to highlight some of the important aspects of the results perceived from other perspectives (e.g. a particular load factor), but also convenient referring to the particular results in the discussion. The organization of information presented in the charts is explained in Figure 66 and the paragraphs that follow.

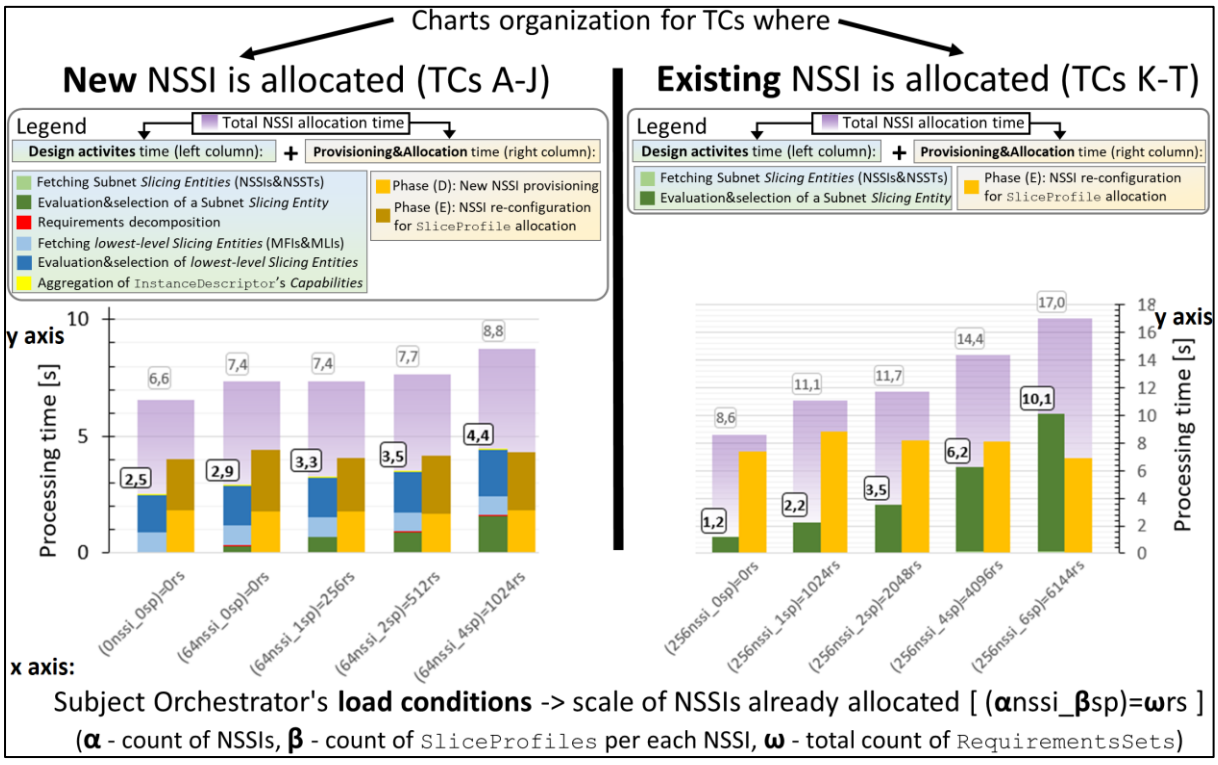


Figure 66. Organization of Charts depicting evaluation results from performance Test Cases in various load conditions.

On the ordinate axis, the measured time in seconds is presented. On the axis of abscissa, different scales of NSSIs already allocated are identified by the ID of the Subject Orchestrator's load conditions, as established

in Table 22. The load IDs are sorted by the NSSIs count (**α**) in the first place and then by the count of `SliceProfiles` assigned to each NSSI (**β**) – such organization divides each chart into 3 clusters of load conditions with constant values of **α** (64, 128, 256). In each of such clusters, the load IDs on the axis of abscissa with a linearly growing total count of `RequirementsSets` (**ω**).

Each of the plotted columns is composed of two internal stacks encompassed by a wide, violet background. The left stack consists of times spent in the design activities (phases A-C of the ASD workflow), whereas the right stack presents the times spent in the provisioning and allocation procedures (phases D-E). The total time spent in the design activities is distinguished in the bolded label on top of the left stack. In turn, the greyed-out label on top of the violet background depicts the total allocation time.

Depending on whether the considered TC results in the allocation of a new NSSI (left part of Figure 66) or an existing NSSI (right part), the two internal stacks contain different ingredients, corresponding to the different paths through the ASD workflow. For the TCs resulting in the allocation of a new NSSI (TCs A-J), the left stack consists of the times spent on: fetching of *Slicing Entities candidates* representing Subnet (light green), their evaluation (dark green), *Requirements* decomposition (including determination of *Capabilities* to be allocated, which reuses the logic of the `decompose` method) (red), fetching of *lowest-level Slicing Entities candidates* (light blue), their evaluation&selection (dark blue), and finally, aggregation of `InstanceDescriptor`'s *Capabilities* (yellow). In the right stack, the times of provisioning and allocation of the newly designed NSSI are presented: the time of phase (D) "New NSSI provisioning" (orange) and phase (E) "NSSI re-configuration for `SliceProfile` allocation" (gold). In the TCs resulting in the allocation of an existing NSSI (TCs K-T), the left column is composed of two ingredients: fetching of *Slicing Entities candidates* representing Subnet (light green) and their combined evaluation&selection (dark green). In turn, the right column represents the time of executing the procedures of phase (E) "NSSI re-configuration for `SliceProfile` allocation" in the selected, existing NSSI (orange).

In the following Subsections, charts visualizing results of TCs execution are presented and commented. The analysis begins by examining the charts for TCs that exercise the ASD workflow for the allocation of **new**, **not-loaded** NSSIs of various scales (TCs A-E). This perspective is then juxtaposed with the results for the allocation of new but **loaded** NSSIs (TCs F-J) and finally with the allocation of **existing** NSSIs (TCs K-T). For the sake of brevity, only selected ones from the total of 20 charts are presented, drawing the reader's attention to the most important aspects.

6.4.5.1. Allocation of new, not-loaded NSSIs (TCs A-E)

Below, Chart 1 and Chart 2 depict results for the allocation of new, not-loaded NSSIs in the smallest and largest scales are presented: TC-A "new.**tiny**.not-loaded" and TC-E "new.**big**.not-loaded". By analyzing their content, several aspects shall be noted as commented in the paragraphs below.

Chart 1. Performance results of executing Test Case A (allocation of new NSSI with 4 not loaded MFIs) in various load conditions.

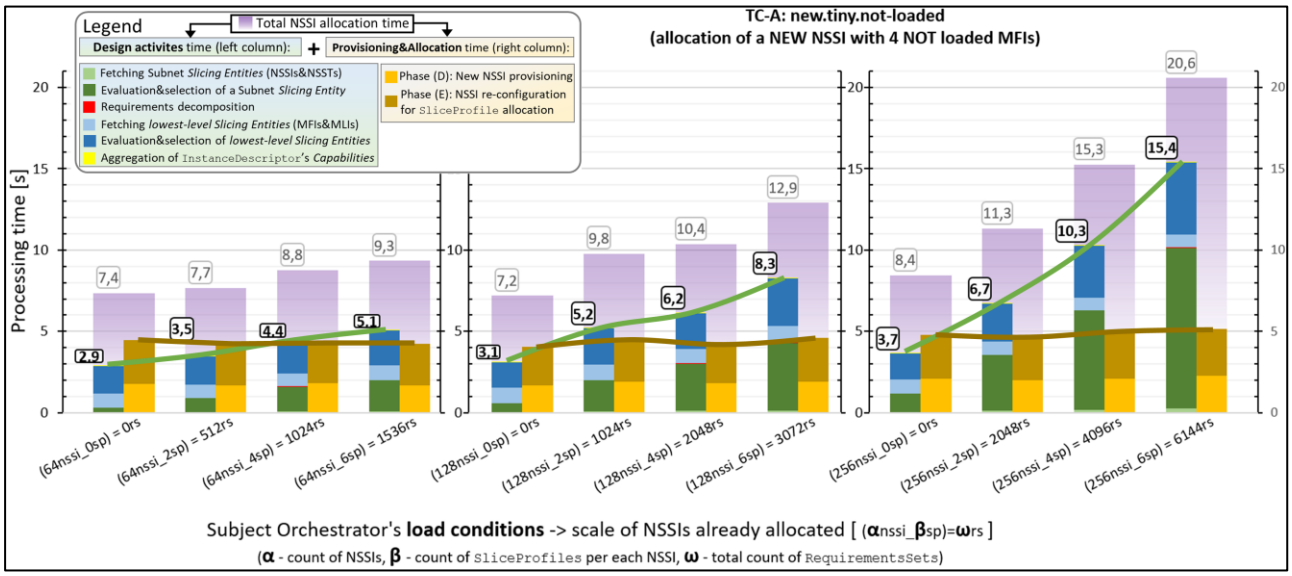
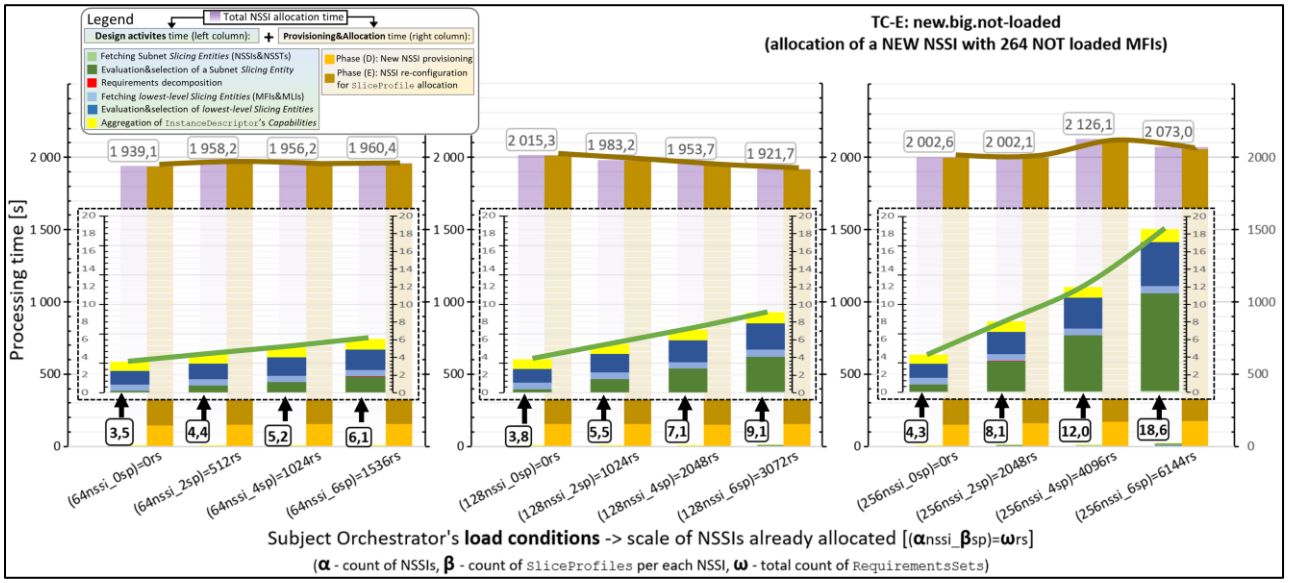


Chart 2. Performance results of executing Test Case E (allocation of a new NSSI with 264 not loaded MFIs) in various load conditions.



Firstly, the **fractions of time consumed** by the particular design activities shall be commented. As visible in both charts, the most of the time in the left columns is consumed on the fetching and evaluation of *Slicing Entities candidates* representing Subnet (NSSIs&NSSTs), indicated with green ingredients, and the *lowest-level Slicing Entities candidates* (MFIs&MLIs), marked with blue ingredients. In both cases the *candidates'* fetching time is linearly dependent on their count. This dependency may not be clearly visible in the charts because all experiments are based on the same arrangement of the gNB NEs data - thus, the MFIs&MLIs *candidates* fetching time (light blue) is constant in all of the considered load conditions. In turn, the fetching time of NSSIs&NSSTs *candidates* is negligible (light green, not even visible) as it is performed locally within the RAN NSSMF component, which hosts the ASD workflow's logic[186]. The MFIs&MLIs *candidates* fetching time is noticeable, because they need to be serialized and transferred from the NFMF to the RAN NSSMF component.

Performance of the evaluation activity (dark green and dark blue) is dependent linearly both on the count of the considered *candidates* and the count of `RequirementsSets` assigned to them. Such relationship can be noticed in each cluster of load conditions and is marked with green trend lines. *Requirements* decomposition times are negligible comparing to the other design activities, which makes them rarely visible in the charts (red color). It is because its implementation at the current stage of ASD PoC is computationally simple - all the elementary calculations dictated by the currently supported `RDRs`[187] can be performed in-memory. In turn, the time of processing in the aggregation of `InstanceDescriptor`'s *Capabilities* activity (yellow) is directly dependent on the scale of the NSSI-to-be-allocated. This dependency follows common sense: for designing a "tiny" NSSI (TC-A), composed of 4MFIs, this processing time is barely visible (<0,1s), whereas when designing a "big" NSSI (TC-E), which contains 264 MFIs, the fraction of time consumed by this activity is noticeable, reaching up to ~1,5s in the most demanding load conditions (256nssi_6sp)=6144rs.

Secondly, it is interesting to note that the total time of designing a "tiny" NSSI is within the range of 2,5 to 15,5 seconds, which is not much different from the results for a "big" NSSI, ranging from 3,5 up to 18,5 seconds depending on the experienced load conditions. By also considering the results from the rest of the TCs in the "new.not-loaded" group (TC-B: 2-15s, TC-C: 4-16s, TC-D: 2,5-16,4s) a conclusion could be drawn that in the given load conditions **the design process takes a similar amount of time regardless of the size of the NSSI-to-be-allocated**. However, as it will be shown in the next Subsection, **such situation is only observed if the designed NSSI is built from not-loaded internals**. It is because although the same count of loaded&not-loaded *candidates* is fetched and evaluated in the given load conditions, which consumes the most of the computation time, only the not-loaded *candidates* are processed in the remaining steps of the workflow,

[186] The fetching times could be higher if the ASD workflow would be implemented inside of a separated, supportive component assumed in the centralized variant of the design process (see Subsection 5.7.3).

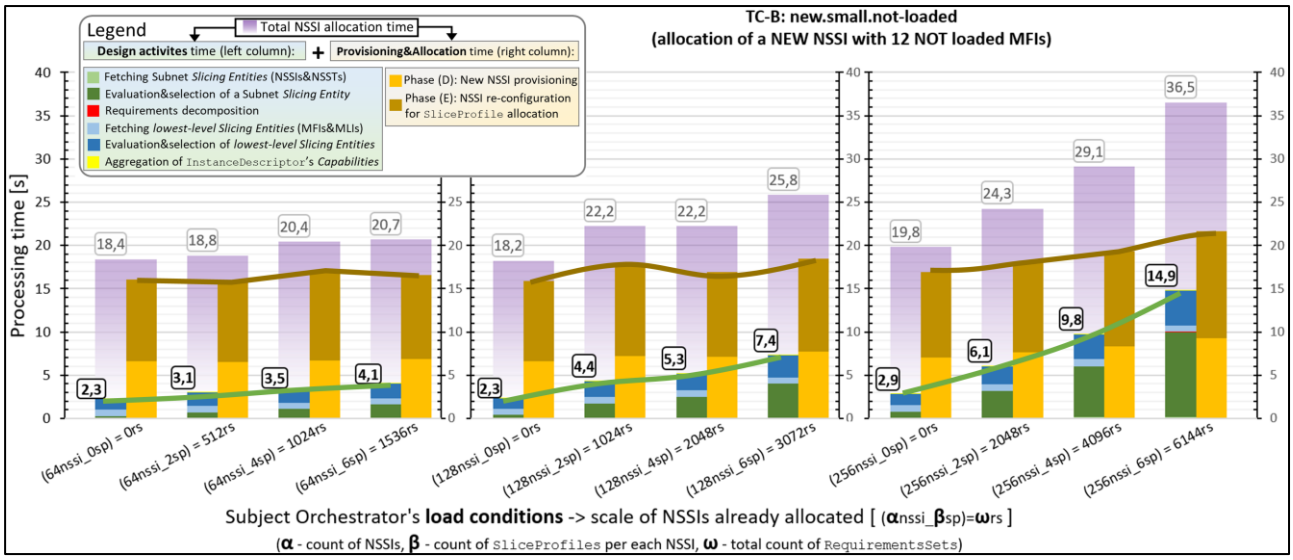
[187] The *Requirements decomposition* processing time could increase if more advanced decomposition strategies would be implemented, e.g. policies based on advanced data analytics, fetching of decomposition decisions from the outside of the Subject Orchestrator, etc.

which takes a tiny fraction of the processing time. Hence, by comparing the charts for TC-A and TC-E, it can be noticed that the difference in the total design time is mainly caused by the calculation of the *aggregated Capabilities* of the created `InstanceDescriptor` (yellow color), which naturally increases with the size of NSSI-to-be-allocated, as discussed in the previous paragraph.

Thirdly, it is worth noting the **difference in the dependency on the load conditions between the design process and the provisioning&allocation procedures** (left column vs right column). As discussed in the paragraphs above, performance of the design activities is linearly dependent on the growing count of NSSIs-already-allocated and the growing count of `SliceProfiles` assigned to them. In contrast, the provisioning&allocation procedures operate only on the *lower-level Slicing Entities* selected in the design process, which in TCs A-E are not-loaded. Hence, the increasing load conditions, which pertain to the "loaded MFIs pool" only, do not have a significant impact on the performance of the provisioning&allocation procedures in the currently considered TCs. As a consequence, the provisioning&allocation time is similar in all load conditions in a given TC (see the brown trend line in the charts). However, it shall be noted that the **provisioning&allocation time rises significantly with the growing scale of the NSSI-to-be-allocated**. In the small to mid-scale the provisioning&allocation times are decent (TC-A with ~4-5s, TC-B with ~16-21s and TC-C with ~90-107s), but with larger scale these times grow to values that can be perceived unacceptable by the Provider and the Customers: TC-D with ~315-385s (~5-6,5min), up to TC-E with ~1800-2110s =~30-35min (see Chart 2). The problems with the performance of the provisioning&allocation procedures implemented in the Subject Orchestrator are discussed further in the upcoming Subsections.

Another observation is that in almost all of the load conditions in TC-A the design activities take more time than the provisioning&allocation procedures. As an example, in the heaviest load conditions (256nssi_6sp)=6144rs the design takes 15,4s whereas the provisioning&allocation takes only 5s (designing NSSI constitutes almost 76% of the total allocation time). However, this is the only TC in which such proportions can be observed. Already in TC-B, which considers a "small" scale Subnet with 12 MFIs, the time spent on the provisioning&allocation procedures overruns the design process time in all load conditions, reaching 14,9s vs 21,6s (41% for NSSI designing) in the most demanding conditions, as visible in Chart 3 below.

Chart 3. Performance results of executing Test Case B (allocation of a new NSSI with 12 not loaded MFIs) in various load conditions.



With the growing scale of NSSI-to-be-allocated the difference between the time spent on the design and provisioning&allocation grows significantly. When considering the heaviest load conditions, in TC-C ("medium") it takes 16,2s for designing and 107,9s for provisioning&allocation (13% for design), in TC-D ("large") it is 16,4s vs 367,2s (4%) and in TC-E ("big") it is 18,6s vs 2056,4s =~ 34minutes (0,9%). As shown, with the growing scale of NSSI-to-be-allocated only a **tiny fraction of time is spent on the design process**. This difference grows even further when the NSSI-to-be-allocated is designed from loaded internals, which is discussed in the next Subsection.

6.4.5.2. Allocation of new, loaded NSSIs (TCs F-J)

While the discussion in the previous Subsection considered the allocation of new NSSIs designed from not-loaded internals, now it is focused on TCs resulting in the allocation of **new** NSSIs that are designed from **loaded** internals. Below, two charts are presented: Chart 4 presents the results for the allocation of a "new.**tiny**.loaded" NSSI (TC-F) and Chart 5 shows results of "new.**big**.loaded" (TC-J) allocation.

Chart 4. Performance results of executing Test Case F (allocation of a new NSSI with 4 loaded MFIs) in various load conditions.

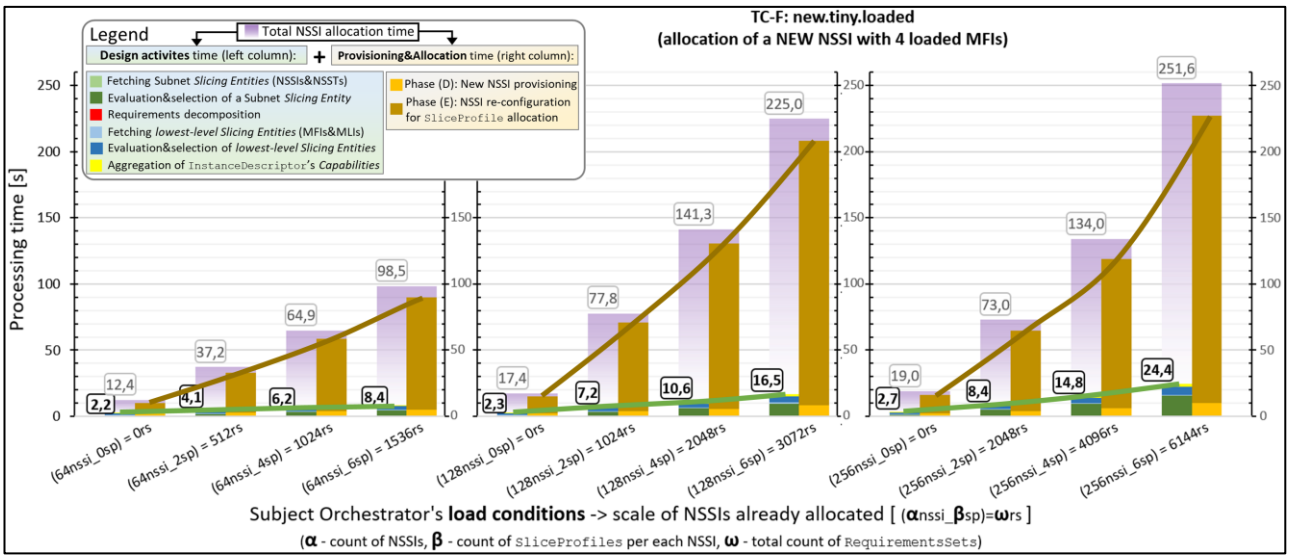
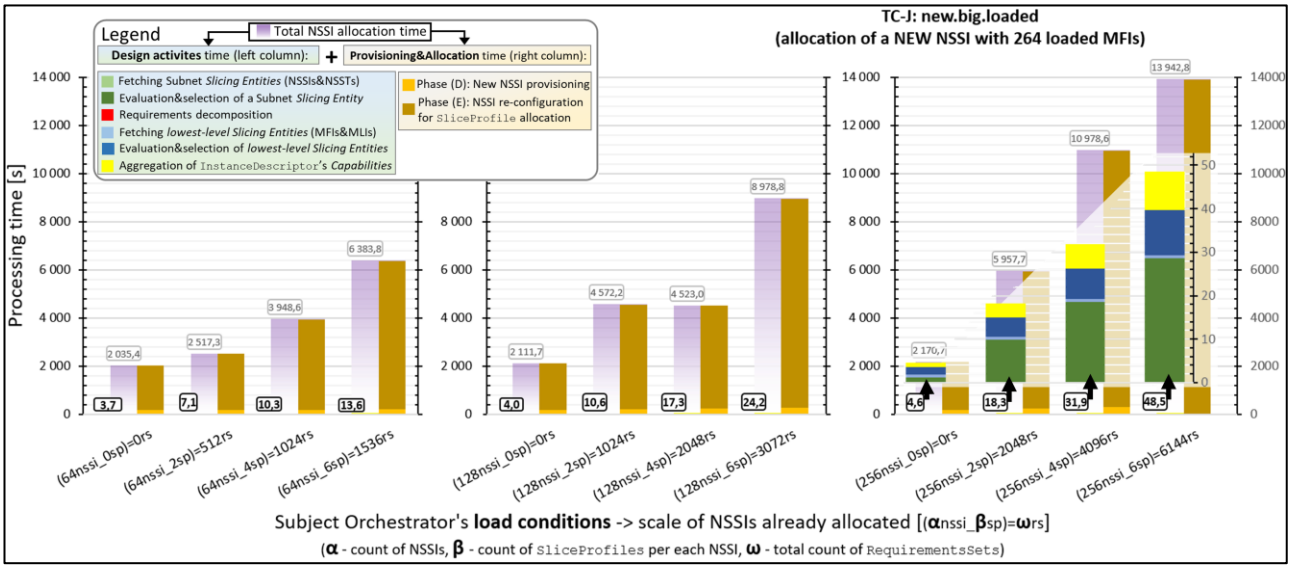


Chart 5. Performance results of executing Test Case J (allocation of a new NSSI with 264 loaded MFIs) in various load conditions.



By analyzing these Charts and the individual results it can be stated again that the **duration of the design process** is **linearly dependent on the count of processed *candidates* and the count of `RequirementsSets` managed in the Subject Orchestrator** – these observations confirm the conclusions discussed in the previous Subsection. In the currently considered TCs, though, the **impact of utilizing loaded MFIs** in the designed NSSI is noticeable, which manifests itself in noticeably longer processing in the evaluation (dark green, dark blue) and aggregation activities (yellow) comparing to the TCs in which the not-loaded MFIs have been utilized. Nevertheless, even in the most demanding conditions in TC-J (allocation of a "big" NSSI, composed of 264 already loaded MFIs) the design process takes ~48s. **Processing time under 1 minute can be considered a satisfactory result**, bearing in mind that the NSSI being allocated encompasses all MFIs in the RAN domain, and the Subject Orchestrator already manages 6144 other Slices at the same time. In summary, the results from TCs F-J demonstrate **satisfactory performance of the design process** and confirm that the **count of the**

already managed `RequirementsSets` [188] constitutes a major factor influencing the design process performance. However, while tweaking the performance of the design process is possible and desirable, these are not the design activities that require the most attention in the end-to-end NSSI allocation.

Sadly, **the satisfactory efficiency of the design process is overwhelmed by the suboptimal performance of the allocation procedures executed in phase (E)**. Although their execution time **grows linearly** with the count of already managed `RequirementsSets` (see brown trend lines), the **steepness of growth is significant**. This is clearly visible by comparing the results of TCs that consider the same scale of NSSI-to-be-allocated, but composed of not-loaded (discussed in the previous Subsection) and loaded MFIs (this Subsection), e.g. "tiny" NSSI in TC-A vs TC-F or "big" NSSI in TC-E vs TC-J. In TC-A, the measured allocation time in all load conditions was approximately the same, oscillating between 4 and 5s. In contrast, the allocation times in TC-F are much larger in terms of absolute values, and they grow significantly with the increasing load conditions: minimally 8s in load conditions (64nssi_0sp)=0rs, maximally 227,2s in load conditions (256nssi_6sp)=6144rs. Such trend is observed in each cluster of load conditions. Similar dependencies and radically growing time of allocation are observable in other TCs from the currently considered "new.loaded" group (TCs G, H, I), with the longest processing times measured for allocating a new "big" NSSI in TC-J - in the most demanding conditions (256nssi_6sp)=6144rs allocation takes up to 13894,2s =~ 231,5min =~ 3h52min (sic!).

A closer analysis of the measured provisioning times and code-level debugging performed by the author reveals the need for major improvements of the allocation procedures implemented in the Subject Orchestrator. Although they are functionally valid, the performance of these procedures is handicapped by the combination of two factors: transactional, sequential application of SLS configurations to MFIs by the NFMF component and repeatable execution of the heavy-weight MFIs' *Capabilities aggregation* activity after each step of the allocation sub-workflows. Although such implementation decisions contribute to high consistency of the data kept in the Subject Orchestrator's inventories, they cause **suboptimal allocation times, which grows greatly with each `RequirementsSet` added to the system**.

The **poor performance of the allocation procedures has a direct, negative impact on the end-to-end NSSI allocation time**. While the allocation of small and mid-scale NSSIs in low load conditions can be perceived decent (e.g. ~93-124s in TC-C), in all other cases the long allocation times translate into **unacceptably slow execution of the entire end-to-end allocation pipeline**, both from the Operator's and Provider's point of view, but above all from the Customer's perspective, who expects prompt allocation of NSSIs. The observations provided by the author provide directions for improvements to be applied by the development team in the next phases of Subject Orchestrator's evolution. However, as numerously underlined, reworking these procedures is not related to the merits of this thesis, as the author focuses purely on the design aspects. Hence, such improvement efforts are out scope of the experimental works conducted within this dissertation.

6.4.5.3. Allocation of existing NSSIs (TCs K-T)

Execution of TCs K-T, due to the allocation of existing NSSIs, investigates a limited fraction of the ASD workflow (phases A, B and E), where only fetching, evaluation and selection of NSSI *candidates* is relevant in the context of the design process. Thus, only selected aspects differentiating the currently considered results from the previously analyzed TCs are briefly commented. By analyzing Chart 6 for TC-N ("existing.medium.not-loaded") and Chart 7 for TC-S ("existing.large.loaded"), several aspects that pertain to the entire group of the considered TCs can be noticed – the observed trends are the same.

[188] `RequirementsSets` count corresponds to the count of `SliceProfiles` managed in the system (which directly corresponds to the count of NSIs currently managed) and the arrangement of internal elements of NSSIs to which those `SliceProifles` are assigned.

Chart 6. Performance results of executing Test Case F (allocation of an existing NSSI with 132 not loaded MFIs) in various load conditions.

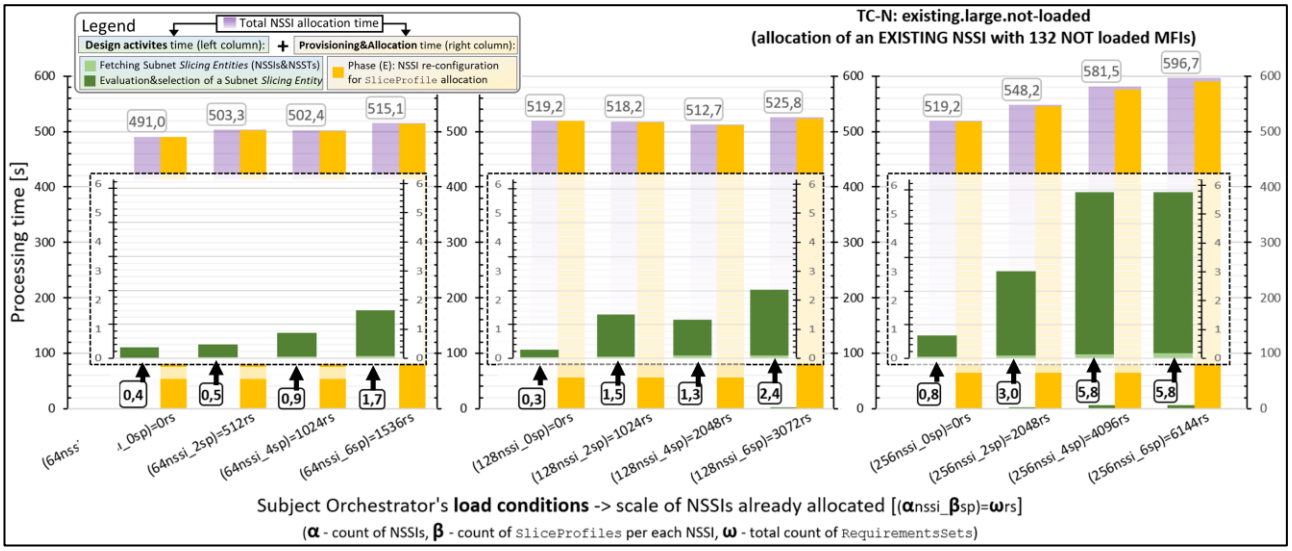
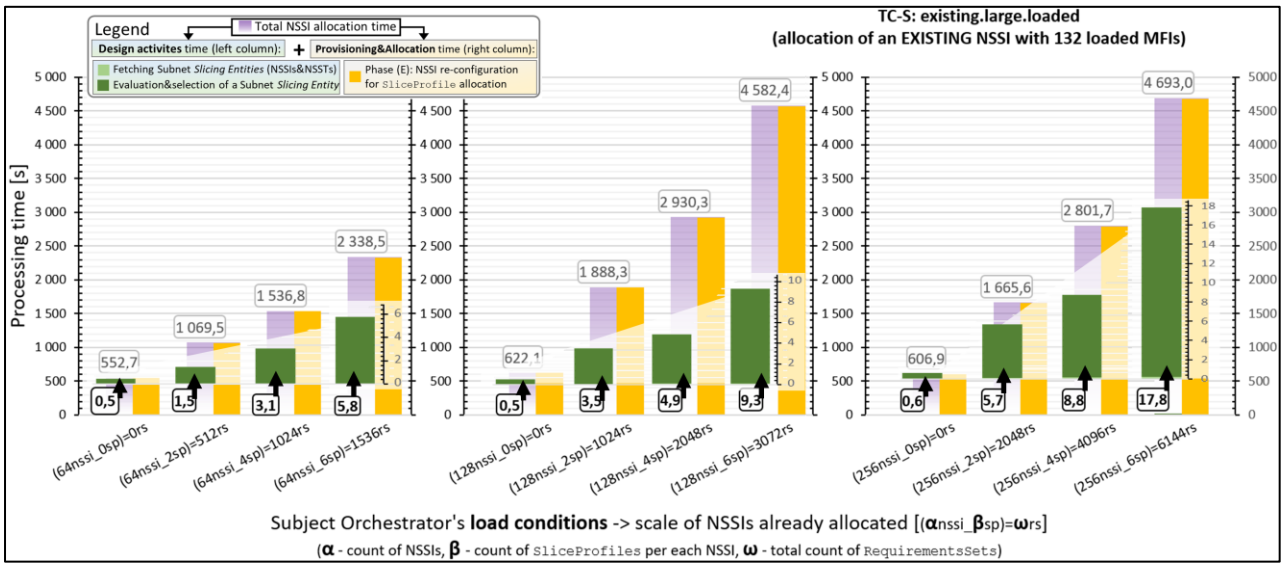


Chart 7. Performance results of executing Test Case S (allocation of an existing NSSI with 132 loaded MFIs) in various load conditions.



The processing time in the design activities in all TCs are satisfactory, reaching 17,8s in TC-S in load conditions (256nssi_6sp)=6144rs. What is worth noting is the effect of the combined implementation of the evalution&selection activities, coupled with the feedback mechanism – thanks to such approach, not all of the NSSI candidates need to be processed, because the evaluation is interrupted as soon as a candidate satisfying the selection criteria is discovered (see Subsection 5.5.3.7 for details). For this reason, the evaluation&selection times are shorter than in the case of the corresponding TCs A-J (see previous Subsections), in which all NSSI candidates have been firstly evaluated before the decision to design a new one was made in the ASD workflow.

The allocation procedures executed in phase (E) suffer from the same deficiencies that have been identified in the previous Subsection. As a result, the dependency of the allocation time on the load of the NSSI being reused and on the experienced load conditions is significant. This manifests itself in decent times of processing for small to mid-sized NSSIs in low to average load conditions, but unacceptably long processing in larger scales. When reusing a not-loaded NSSI (TCs K-O) the allocation time grows with increasing load conditions, but the rise is not severe, i.e. ~5-8s for “tiny” (TC-K), ~40-53s for “small” (TC-L), ~122-163s for “medium” (TC-M), ~490-590s for “large” (TC-N) and ~1820-2120s for “big” (TC-O). However, in the case of loaded NSSIs (TCs P-T), the allocation time increases drastically with the more demanding load conditions. This causes unacceptably slow end-to-end allocation already for mid-scale NSSIs in average load conditions, e.g. it takes 371s for reusing a “medium” loaded NSSI (TC-R) in load conditions (128nssi_2sp)=1024rs. The

longest end-to-end allocation time has been measured for reusing a "big" loaded NSSI (TC-T) in load conditions (256nssi_4sp)=4096rs, which lasted 17 386s =~4h50min[189].

6.4.6. Performance evaluation conclusions

Based on the results obtained from the conducted experiments, it can be stated that the **performance of the automated design activities executed in phases A, B and C**, which reaches ~48s in the most demanding of the exercised circumstances, is **fully satisfactory** for supporting prompt allocation of **both existing and newly designed NSSIs, regardless of their scale and** the **load conditions** experienced by the Subject Orchestrator. This observation makes the elaborated concepts and their PoC implementation adequate for supporting the on-demand NSSIs allocations. Offering "dynamic slicing" management capabilities based on the delivered mechanisms also becomes possible. Thus, the efficiency of the author's implemented PoC solution elements meets the performance-related non-functional requirements stated in Section 2.4 fully.

In turn, the current implementation of the provisioning&allocation procedures executed in phases D and E exhibit decent performance in small scale of NSSI-to-be-allocated and in low load conditions, making them sufficient for supporting end-to-end, on-demand NSSIs allocation in small scales. However, significant performance improvements need to be applied in the next phases of Subject Orchestrator's development in order to support larger scales of NSSIs, both to-be-allocated and the already-allocated ones. These improvements need to be applied in order for the Subject Orchestrator to support efficient, end-to-end on-demand NSSIs allocation and their management according to the "dynamic slicing" principles.

[189] Attempt of allocating a "big" loaded NSSI in load conditions `(256nssi_6sp)=6144rs` failed due to timing-out the 5h limit established by the author as a maximum TC execution time.

Chapter 7

Conclusions

The results and observations presented in the previous Chapters were concentrated on a range of aspects of the proposed solution, discussed in specific, narrowed contexts. In turn, this Chapter takes a global outlook on the collected results, consolidating the previously formulated partial conclusions into conclusions of general value. Firstly, the conclusions important for verifying the thesis statements are discussed. Then, the contributions of this dissertation are assessed from the potential of their industrial adoption. Finally, directions for future research and development are outlined, including possible extensions of the conceptual framework and its PoC implementation, as well as discussion of the currently observed and anticipated trends that have potential of influencing the Network Slicing design-time area in the near future.

7.1. Thesis verification

The thesis formulated in Section 1.4 expects the proposition of a conceptual framework that could be utilized for enabling automated Network Slices/Subnets design performed in a declarative manner - that is, by automatically transforming the high-level, Customer-perspective SLS *Requirements* into a low-level technical specification of an NSI/NSSI that would meet them. The exact wording of the thesis is repeated below:

With the synergy of modeling the Operator's and Provider's domain, operational and business knowledge, uniform modeling of the Network Slice/Subnet building blocks and the structuralization of the design activities to leverage the modeled knowledge, it is possible to define a coherent and generic conceptual framework of a design process that enables construction of multivariant and extendable systems performing automated transition from high-level SLS requirements to a technical specification of an NSI/NSSI capable of meeting them.

To effectively solve the stated problem, the author proposed a broad set of conceptual models and methods that provide adequate foundations for constructing design automation solutions exhibiting the stated features. Among multiple insights revealed while elaborating these concepts, but also while building their PoC implementation and during the conducted empirical experiments, the gathered results allow for formulating conclusions that directly pertain to the thesis statements - the reasoning below provides the arguments to justify their validity.

The most important arguments confirming the thesis statements are provided in Section 6.3.5, which discusses the conclusions from the functional evaluation of the PoC solution. The conducted experiments allowed for **empirically verifying** that the ASD workflow built according to the concepts of the proposed framework **offers the desired functionalities** and **delivers the expected results**, proving its **functional validity**. Thanks to these observations, the core statements of the thesis can be confirmed, namely that the proposed framework is **feasible** and a solution built in accordance with its concepts **indeed enabled automation** of the design process, which, in turn, allowed for **excluding the human Operator's participation** in handling the allocation requests triggered by the Customers.

Several aspects featured in the proposed framework were key to achieving this enablement. First of all, the framework leverages a **modeling approach** that enforces **representation** of the Operator's and Provider's **knowledge** in **well-structured, generic models**, as well as **uniform representation** of the **Network Slice/Subnet building blocks as encapsulated units with exposed *Capabilities***. These modeling decisions are crucial for **enabling easy programmatic processing** of the elements comprising the designed Slice/Subnet and for exploitation of the captured knowledge in an automated design process. In practice, the pointed virtues are clearly visible in a range of models implemented in the ASD PoC that correspond to their

conceptual counterparts, especially in the construction of models representing the SLS *Requirements* and *Capabilities*, the hierarchy of *Slicing Entities*, abstract Subnet's structure templates (`NSST` with `AbstractNodes`) and the `RDRs`, as justified in the paragraph below.

The uniform representation of *Requirements* and *Capabilities* as `SlsAttributes` containing specification of their **behaviors** is an **important enabler for automated processing** of such entities in the course of the design activities (automated evaluation, aggregation, etc.). Next, the enforced exposition of *Capabilities* by every *Slicing Entity* is crucial for **enabling automated and uniform matching** of heterogeneous *candidates* to the posed *Requirements*. Further, the definition of `RDRs` **enables automated transformation** of the customer's *Requirements* into lower-level *Requirements* against the abstract elements of the designed Subnet (i.e. `AbstractNodes` in the `NSST`). This, in turn, **enables their automated substitution** into concretized elements that meet the posed *Requirements*. As shown, it can be concluded that **appropriate modeling serves as a direct enabler for achieving automated processing** in the individual design activities.

Secondly, the proposed framework enforces **structuralization of the design process** into a set of coherent, fine-grained, single-responsibility individual activities with clear inputs-outputs and complementing roles. Such functional organization enables **flexible construction** of systems that deliver the desired functionality in a selected **variant**, depending on the architectural assumptions and functional scope of the Orchestrator being augmented, as shown in the ASD PoC incorporated into the Subject Orchestrator. Importantly, the assumed structuralization of the design activities **leverages the knowledge** conveyed in the models discussed above, which is essential for **integrating the Operator's and Provider's domain and business expertise** into the design pipeline and for enabling automated processing of the allocation requests in a variant that follows the operational and business intentions. Most notably, this is shown in the ASD PoC, where the execution of the particular variant of the design activities' flow is steered by the configuration of the aforementioned models and the `DesignPolicies` that can be flexibly decided by the Operator and Provider, as agreed with the Customer.

Finally, Subsection 5.7.2 provided a range of conclusions that confirm a number of technical characteristics of the proposed framework claimed in the thesis, in particular its **feasibility**, **coherency**, **multivariance**, **genericity**, **flexibility** and **extendibility**. Thanks to these virtues, not only the proposed framework is open for extensions, but the solutions built according to its concepts exhibit high **structural scalability**[190]. This enables expanding the operational scope of the developed solutions without major changes in their architecture, by reusing the already implemented generic models and mechanisms.

The entire reasoning and argumentation presented above allows the author to conclude that the **thesis has been verified successfully**.

7.2. Discussion on the industrial applicability of the proposed concepts and their potential impact

The discussion in this Section attempts to answer whether the elaborated concepts and their technical realization can find practical application in the industrial reality. Many of the discussed observations and results obtained in this dissertation give such hope. The main reason for claiming so is because the utilization of the proposed mechanisms can bring **tangible technical and business benefits** for both the Providers and their Customers, as noticed at various stages of this thesis and as summarized below.

Automation of the design process brings a radical change to the management of Network Slices/Subnets, which in orchestration solutions that lack such functionality have to be established with long-lasting preparations by experts – this pain point, as well as several others mentioned below, are resolved by the solutions proposed in this thesis. Thanks to the automated design, the **human Operators' involvement in the management duties is reduced**, because the Orchestrator processes the incoming allocation requests **autonomously** - it is the Orchestrator's duty to dynamically decide on the utilization of available resources for the construction of suitable NSIs/NSSIs by following the `DesignPolicies` specified by the Operator and Provider. Moreover, due to the fact that there is no need to wait for the human Operator's intervention, the

[190] Examples proving structural scalability of the PoC solution which implements the conceptual framework are provided in Subsection 5.7.2.

ordering and **delivery of an NSI/NSSI is prompt**. Hence, Provider benefits from **OPEX savings**, **increased productivity** of its staff and more **effective exploitation** of the managed infrastructure.

Furthermore, automation of the design process enables the exposition of a *declarative provisioning style* API, which, by combining it with the support of SLS *Requirements* exhibiting high-level semantics[191], brings a significant value not only for the Provider but also to its Customers. It is because allocating an NSI/NSSI, unlike in the *imperative style* solutions, **does not require making any design-related decisions and possessing any low-level telco domain knowledge**. The client of such API can express its demands utilizing tangible, high-level SLA/QoS measures oriented on the Service that will run on the requested Slice. This means that triggering of NSI/NSSI allocation becomes **easy** and **can be performed even by non-experts**.

By leveraging these features, the Providers can establish **new business offerings**, like on-demand spawning of Slices/Subnets offered in the "Network Slice/Subnet as a Service" mode. Such Slices/Subnets may be established programmatically **by the Customers themselves** and be utilized for various use cases that were not possible until now, e.g. spawning short-lived, small-scale Slices for smooth gaming or teleconferencing experience, etc. As such, automated design combined with the declarative API provide a powerful tooling that paves the way towards flexible and dynamic creation of Slices/Subnets in potentially massive quantities, spawned not only by "big" Customers (i.e. vertical industries, corporations, etc.), but **even by individuals**. The author believes that this is an important feature of the elaborated concepts that may contribute to a more **widespread adoption** of the Network Slicing technology and its **ubiquitous exploitation**.

Already the factors mentioned in the paragraphs above provide a strong blend of business and technical arguments that justify the consideration of industrial application of the elaborated concepts. Another factor that is frequently mentioned as decisive on the adoption of a given system in the telco industry is its **standards-compliancy**. Standard-compliancy is of fundamental importance for the Providers, as it largely increases the chances of utilizing solutions that provide the desired functionality and integrate well with the other systems, decreasing the risk of Vendor lock-in. In this respect, the conceptual framework elaborated by the author and the solutions built in accordance to it, are **fully aligned** to the latest versions of the Network Slicing standards in force, in particular with the provisioning APIs and models enforced by the 3GPP and ETSI specifications. Therefore, this aspect also speaks in favor of the proposed concepts in terms of their industrial application.

Although all the arguments provided above are supportive, the potential adoption of the presented concepts is inextricably dependent on the commercial success of the Network Slicing technology in general. This factor is raised, because while named as an essential feature of 5G, the adoption of Network Slicing has been progressing slowly at the time of finishing this dissertation, despite the fact that it has been developed already for several years. There are not isolated fears that this technology currently does not have a "killer application" and might not find enough interest from the Customers to economically justify its adoption in the 5G timeline.

While the author partially shares these concerns, there are numerous business reports that forecast a significantly rising value of the market related to Network Slicing in the upcoming years [22], [170], [171]. Moreover, the 6G visions assume Network Slicing to be one of its essential building blocks and heavily rely on its availability for the realization of the envisioned use cases [14]–[17]. Thus, the author believes that, ultimately, the continuous progress in the telco industry will force Providers to offer Slices as part of their commercial offerings, and the need of spawning their increasing quantities will emerge. The author expects that just like spawning a Virtual Machine or a container is nowadays perceived as an obvious way of establishing the compute, storage and localized network connectivity, spawning a Network Slice in the 5G-Advanced and 6G networks will become the basic and lightweight method of establishing on-demand, wide-area telecommunication infrastructure profiled for supporting a large variety Services, including the highly-specific and demanding ones. At its target maturity, the capabilities of establishing massive quantities of on-demand NSIs/NSSIs should be achievable, which will result in the possibility of allocating a dedicated Slice for

[191] i.e. `dLLatency`, `uLLatency`, `dLThptPerSlice`, `uLThptPerSlice`, `maxNumberofUEs` supported in the PoC, or others.

each Service. In order to embody such visions, **full automation of the design-related procedures is indispensable**, which seals the need for utilizing the concepts presented in this dissertation or similar.

The combination of arguments discussed above clearly illustrates that the contribution of this dissertation was not only sufficient for the successful verification of the scientific thesis statement but it constitutes a significant improvement to the state-of-the-art in the domain of Network Slicing design-time aspects and has high potential for being industrially adopted.

7.3. Future work

Although the proposed conceptual framework already covers a wide range of aspects in the Network Slice/Subnet design-time area, it is ready to be expanded. As an example, currently the **SLA→SLS translation** is not within the scope of the framework, which is mainly due to the immaturity of the corresponding standards that prohibited the author from elaborating methods for performing and automation of such translation in a credible manner[192]. However, with the growing maturity of the 3GPP and GSMA specifications, inclusion of the SLA→SLS transformation might become a part of the framework. In particular, once the mapping between the attributes contained in these data structures becomes unequivocal, similar methods and models as the ones utilized for the automation of the *Requirements decomposition* activity (i.e. `AbstractNodes`, `RDRs`, etc.) might be leveraged.

The area that is currently not thoroughly covered by the proposed conceptual framework is the **organization of the design process within the Transport Network** (TN) and **Data Network** (DN) domains, lack of which was already mentioned in the dissertation's scope definition in Section 1.6. Although several aspects relevant for the TN domain have been touched upon in this thesis, their comprehensive elaboration in alignment to the recently emerging IETF standards, which begun to take mature shape already after finishing the conceptualization phase in the author's research[193], constitutes an important subject of future work. Thus, further generalization of the proposed concepts and their adaptation towards supporting TN and DN domains' particularities will constitute another step towards comprehensive coverage of the design-time aspects across all telco technological domains.

Another aspect that is considered by the author as a future work, is the generalization of the framework's concepts towards **supporting other inputs to the design process** than the high-level, Customer and Service-oriented SLA/SLS *Requirements*. As discussed in Chapter 1, the main motivation of the author's research from its beginning few years back, was to solve a particular problem – namely to provide a generic solution that would enable automated processing behind the `allocateNsi(ServiceProfile)` and `allocateNssi(SliceProfile)` operations which have started to be standardized. While the proposed conceptual framework allows for constructing systems that effectively solve this problem, other innovative approaches towards telecommunication networks management in general, and Network Slicing in particular, have been emerging in the meantime. Example of an approach that became a subject of a significant interest in the research and standardization communities in the recent years is the concept of **intent-based orchestration**, which has already been referred to in Subsection 2.5.3. Designing NSIs/NSSIs in the intent-based manner seems to be a generalization of the SLS-based design elaborated in this thesis, taking into account that both of these approaches represent the *declarative provisioning style*. However, due to the fact that the intents may target aspects at various levels of abstraction (not only high-level SLA/SLS characteristics), such approach may decrease the complexity of designing NSIs/NSSIs with highly-specific configurations, which would typically need to be performed in an imperative manner due to the fact that it is difficult to express such highly-specific particularities by means of high-level SLA/SLS *Requirements* only. Hence, further generalization of the proposed concepts is considered by the author, in order to **extend the supported inputs to intents with various levels of abstractions**, including the ones with lower-level semantics.

[192] As discussed in Subsection 2.3.2, currently the mapping between the GSMA's GST and the 3GPP `ServiceProfile` is not comprehensive and can result in ambiguous SLS specifications.

[193] See the dissertation scope in Section 1.6 for the timeline of the author's research with relation to the standardization progress.

Other extensions to the proposed conceptual framework are possible as well. However, its core elements seem to be annealed enough to constitute a reference for implementing standards-aligned Network Slicing design-time automation solutions that can be gradually expanded in long term in a backward-compatible manner. In terms of the PoC future development, multiple potential directions of its extensions have been discussed in Subsection 5.7.3.

7.4. Anticipated research directions in the Network Slicing design-time area

After several years of incubation, Network Slicing technology is now at the frontier of its initial commercial rollouts. This, however, does not mean that the research related to this technology, and its design-time aspects in particular, can be considered finalized. Several research directions that have potential of influencing this area, both at the conceptual and implementation levels, are briefly outlined below.

As already mentioned at the previous stages of this dissertation, intent-based approach is currently one of the most intensively explored directions in terms of telco networks management improvements. This fact is confirmed by the increasing number of publications in this area and the already ongoing standardization activities aiming at establishing common conceptual and technical foundations regarding this approach. Further exploration of this direction may enhance the agility of NSI/NSSI allocation from the Customers' but also Provider's and Operator's perspectives.

Among the publicly available research initiatives, the Nephio project [172] deserves a special attention, as it proposes a promising declarative orchestration approach that is not constrained by the prevailing telco standards. In contrast, Nephio at its basic operational philosophy leverages the well-annealed concepts of the Kubernetes platform, which in the IT domain is already perceived as a de-facto standard of IT workloads orchestration. With this basic assumption, all the *Slicing Entities* are expected to be modeled as Kubernetes Custom Resources (CR) that are individually and collectively managed by means of dedicated Kubernetes controllers that deliver the localized and global control-loops, providing active reconciliation and service assurance. While Nephio does not provide any equivalents to the 3GPP's `ServiceProfile`/`SliceProfile` data structures or the `allocateNsi`/`allocateNssi` operations, allocation of new resources is performed declaratively by following the GitOps principles [173]. Such approach assumes pushing the definition of a requested resource into the GIT repository, which is followed by its instantiation by means of the corresponding Kubernetes controller that supervises resources of a given type over their lifecycle. Exploration of the Nephio project in terms of automated declarative Network Slices/Subnets design and runtime assurance is an area that is expected by the author to receive high attention in the nearest future. It is claimed so, because Nephio open-source community boosted its development efforts, which already resulted in the fourth release of the project in February 2025 – this proves high interest of the telco community in this initiative.

Last, but not least, the author needs to mention the exploitation of the AI in the telco domain, which is among the most intensively investigated aspects in the ongoing research towards 5G-Advanced and 6G, including the corresponding standardization activities in multiple bodies[194]. Incorporation of AI techniques in the operation of Network Slicing Orchestrators constitutes a natural step for optimizing the decisions made in various operational contexts, including the design-time aspects. AI models and methods are expected to be applied at the level of individual design activities but also at the overall design process level. Several examples include collecting more accurate information about *Slicing Entities* (e.g. their continuous scoring, more accurate determination of their current and predicted *Capabilities* values), their AI-assisted selection, but also determination of optimized SLS-assurance behaviors of the designed NSI/NSSI, etc. Especially, thanks to the Machine Learning techniques, AI may largely contribute to the self-adaptation of the mechanisms performing automated design. While possibility of leveraging AI is assumed in the elaborated conceptual framework (in particular, as a model conveying operational knowledge that steers the design activities, but also as a way of performing the closed-loop feedback activity assumed in <AS4>), no detailed elaboration of this topic and no practical embodiment of AI methods in the design process was considered by the author in the scope of the PoC. It is because the performed research aimed at establishing foundations that enable

[194] among others, O-RAN [206] or ETSI Experiential Networked Intelligence (ENI) [207] standardization works.

building functionally working design automation systems which, at the later stages, can be equipped with the AI-supported mechanisms. Such enhancements, however, deserve a dedicated, detailed research to be performed prior their inclusion.

As a final remark, it shall be noted that the directions described above have a common denominator – all of them assume declarative style of Network Slicing orchestration and put special emphasis on **domain and operational knowledge modeling and exploitation**. This fact confirms the legitimacy of the research performed by the author in the recent years. The author believes that thanks to the presented conceptual framework and its comprehensive elaboration, a valuable contribution to the state-of-the-art in this area has been provided, and the elaborated deliverables can be utilized as foundations for further research in this interesting field.

Appendix: Organization of the 3GPP and ETSI NFV-MANO standards

Every SDO has its methodology of organizing the standards it develops. 3GPP and ETSI NFV specifications are grouped into so-called **Releases** - tagged revisions of interrelated and mutually compatible specifications that describe all aspects of a telco system with a given set of features, elaborated in a scope agreed in a particular Release. A given generation or its major advancement (like 5G, 5G-Advanced, etc.) typically span multiple 3GPP and ETSI Releases, as depicted in Figure 67. At the time of work on this dissertation, 3GPP Releases 17-19 and ETSI Releases 3-5 have been considered by the author.

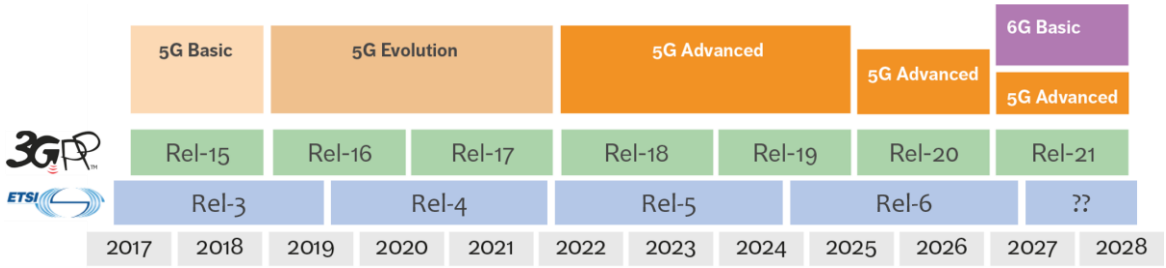


Figure 67. Timeline of 3GPP and ETSI standardization releases against the backdrop of 5G and 6G development schedules.

Within a given Release, 3GPP specifications are divided into **Technical Reports (TR)** and **Technical Specifications (TS)**. TRs typically contain early elaborations of the particular areas or aspects, providing summary of the state of the art, specifying high-level goals, use-cases and requirements as well as proposing early concepts and recommendations for solutions. It shall be underlined that TRs are **non-normative**, which means that their content does not define a standard, but rather a first step towards producing it, constituting and important input for producing TSs. TSs, in contrast, contain **normative** content, including agreed definition of requirements, use-cases, models, procedures, APIs and protocols. Similar division is present in the ETSI specifications, which are divided into TRs and **Group Reports (GR)** that provide informative content, as well as TSs and **Group Specifications (GS)** with normative content.

In the construction of TSs, 3GPP follows a **three-stage methodology** defined by ITU-T Recommendation I.130 [174]. TSs that define the concepts, use-cases and high-level requirements are referred to as Stage 1. Stage 2 TSs, provide so-called Information Sets (IS) which contain technology-agnostic, protocol-agnostic models and APIs that represent the concepts introduced in Stage 1 specifications. All the content in Stage 2 TSs is provided in a vendor-neutral manner by utilizing the constructs of the Unified Modeling Language (UML), which makes them generic and applicable to a wide range of solutions. Finally, Stage 3 TSs, also referred to as Solution Sets (SS), provide technology-specific/protocol-specific modeling of aspects defined in Stage 2 specifications. As an example, the APIs and models defined in Stage 2 with the implementation-agnostic UML constructs are provided in Stage 3 as RESTful APIs, YANG models or by means of other technologies and protocols. Similar approach towards organizing TSs and GSs is adopted by ETSI, by dividing the produced specifications into those defining concepts, requirements and high-level architectures, which are referred to as “Interfaces and Architecture” (IFA) specifications, and their more detailed counterparts that contain concretized, protocol-specific definitions of APIs, models and procedures, which are referred to as “Solutions” (SOL) specifications.

3GPP and ETSI also define guidelines that dictate the rules for the production of TSs. As an example that is relevant for further discussions, a set of standardized terms utilized for referring to the particular concepts is defined in TS 21.905 [79] named “Vocabulary for 3GPP Specifications”. As yet another example, 3GPP utilizes a modeling scheme defined in TS 32.156 [175], which specifies the basic notation and adaptation of the UML constructs to allow for formalized expression of aspects particular to the telecommunications domain. ETSI defines similar rules of UML modeling in GR IFA 017 [176].

List of Acronyms

1G	*1st generation mobile network*	**IS**	*Information Set*	**PRB**	*Physical Resource Block*
4G	*4th generation mobile network*	**IT**	*Information Technology*	**QoE**	*Quality of Experience*
5G	*5th generation mobile network*	**ITU**	*International Telecommunication Union*	**QoS**	*Quality of Service*
6G	*6th generation mobile network*	**KPI**	*Key Performance Indicator*	**RAN**	*Radio Access Network*
3GPP	*3rd Generation Partnership Project*	**LAN**	*Local Area Network*	**RDR**	*Requirement Decomposition Rules*
AI	*Artificial Intelligence*	**LCM**	*Life-Cycle Management*	**RIC**	*Radio Intelligent Controller*
AMF	*Access and Mobility Function*	**MAC**	*Medium Access Control*	**RLC**	*Radio Link Control*
AN	*Access Network*	**MANO**	*Management and Orchestration*	**RP**	*Reference Point*
API	*Application Programming Interface*	**MCC**	*Mobile Country Code*	**RRC**	*Radio Resource Control*
AR	*Augmented Reality*	**MDAF**	*Management Data Analytics Function*	**RRM**	*Radio Resources Management*
ASD	*Automated RAN Design Process*	**ME**	*Managed Element*	**RU**	*Radio Unit*
AUSF	*Authentication Server Function*	**MF**	*Managed Function*	**SA**	*StandAlone mode*
BPMN	*Business Process Model and Notation*	**MFI**	*Managed Function Instance*	**SAP**	*Service Access Point*
BSS	*Business Support System*	**mIoT**	*massive Internet of Things*	**SAPD**	*Service Access Point Descriptors*
CAPEX	*capital expenditures*	**ML**	*Managed Link*	**SBA**	*Service-Based Architecture*
CAR	*Capability Aggregation Rules*	**MLI**	*Managed Link Instance*	**SBI**	*Service-Based Interface*
CCM	*CIS Cluster Management*	**MNC**	*Mobile Network Code*	**SBMA**	*Service-Based Management Architecture*
CDN	*Content Distribution Network*	**MNO**	*Mobile Network Operator*	**SD**	*Slice Differentiator*
CIR	*Container Image Registry*	**MSCS**	*Multi-Site Connectivity Service*	**SDO**	*Standards Development Organizations*
CIS	*Container Infrastructure Services*	**MVNO**	*Mobile Virtual Network Operator*	**SIM**	*Subscriber Identity Module*
CISM	*Container Infrastructure Service Management*	**NBI**	*NorthBound Interface*	**SL**	*Scaling Level*
CM	*Configuration Management*	**NE**	*Network Element*	**SLA**	*Service-Level Agreeement*
CN	*Core Network*	**NEF**	*Network Exposure Function*	**SLAD**	*SLA Descriptor*
CNF	*Containerized Network Function*	**NEST**	*Network Slice Type*	**SLS**	*Service-Level Specification*
COTS	*commodity of the shelf*	**NF**	*Network Function*	**SMF**	*Session Management Function*
CP	*Control Plane*	**NFMF**	*Network Function Management Function*	**S-NSSAI**	*Single Network Slice Selection Assistance Information*
CS	*Communication Service*	**NFV**	*Network Functions Virtualization*	**SOL**	*Solutions*
CSAR	*Cloud Service Archive*	**NFVO**	*Network Functions Virtualization Orchestrator*	**SON**	*Self-Organizing Networks*
CSC	*Communication Service Customer*	**NIC**	*Network Interface Card*	**SRIOV**	*Single Root Input-Output Virtualization*
CSI	*Communication Service Instance*	**NOP**	*Network Operator*	**SS**	*Solution Set*
CSMF	*Communication Service Management Function*	**NRF**	*Network Repository Function*	**SST**	*Slice/Service Type*
CSP	*Communication Service Provider*	**NRM**	*Network Resources Model*	**TA**	*Tracking Area*
CU	*Centralized Unit*	**NS**	*Network Service*	**TAC**	*Tracking Area Code*
CUPS	*Control User Plane Separation*	**NSA**	*Non-StandAlone mode*	**TAI**	*Tracking Area Identity*
DAF	*Data Analytics Function*	**NSaaS**	*Network Slice as a Service*	**TC**	*Test Case*
DC	*Data Center*	**NSD**	*Network Service Descriptor*	**TN**	*Transport Network*
DCI	*Data Center Interconnection*	**NSI**	*Network Slice Instance*	**TR**	*Technical Report*
DF	*Deployment Flavor*	**NSLD**	*Network Slice Descriptor*	**TS**	*Technical Specification*
DL	*downlink*	**NSMF**	*Network Slice Management Function*	**UAV**	*Unmanned Aerial Vehicle*
DN	*Data Network*	**NSSaaS**	*Network Slice Subnet as a Service*	**UDM**	*User Data Management*
DPI	*Deep Packet Inspection*	**NSSI**	*Network Slice Subnet Instance*	**UDSF**	*Unstructured Data Storage Function*
DU	*Distributed Unit*	**NSSMF**	*Network Slice Subnet Management Function*	**UE**	*User Equipment*
EMS	*Element Management System*	**NSST**	*Network Slice Subnet Template*	**UL**	*uplink*
EPC	*Evolved Packet Core*	**NST**	*Network Slice Template*	**UML**	*Unified Modeling Language*
ETSI	*European Telecommunications Standards Institute*	**NWDAF**	*Network Data Analytics Function*	**UP**	*User Plane*
FCAPS	*Faults, Configuration, Accounting, Performance, Security*	**ONAP**	*Open Network Automation Platform*	**UPF**	*User Plane Function*
FM	*Fault Management*	**OPEX**	*operational expenditures*	**URLLC**	*Ultra-Reliable and Low-Latency Communication*
FNO	*Fixed-Line Network Operator*	**OS**	*Operating System*	**V2X**	*Vehicle-to-Everything*
FVNO	*Fixed-line Virtual Network Operators*	**OSM**	*Open Source MANO*	**VIM**	*Virtualized Infrastructure Manager*
gNB	*Next Generation Node B*	**OSS**	*Operation Support System*	**VL**	*Virtual Link*
GR	*Group Report*	**PaaS**	*Platform as a Service*	**VLD**	*Virtual Link Descriptor*
GS	*Group Specification*	**PCF**	*Policy Control Function*	**VM**	*Virtual Machine*
GSM	*Global System for Mobile Communication*	**PDCP**	*Packet Data Convergence Protocol*	**VNF**	*Virtualized Network Function*
GSMA	*GSM Association*	**PDU**	*Protocol Data Unit*	**VNFC**	*Virtualized Network Function Component*
GST	*Generic Slice Template*	**PHY**	*Physical Layer*	**VNFD**	*Virtualized Network Function Descriptor*
GUI	*Graphical User Interface*	**PLMN**	*Public Land Mobile Network*	**VNFFG**	*VNF Forwarding Graph*
IETF	*Internet Engineering Task Force*	**PM**	*Performance Management*	**VPN**	*Virtual Private Network*
IFA	*Interfaces and Architecture*	**PNF**	*Physical Network Function*	**VRAN**	*Virtualized Radio Access Network*
IL	*Instantiation Level*	**PNFD**	*Physical Network Function Descriptor*	**WAN**	*Wide Area Network*
IMT	*International Mobile Telecommunications*	**PoC**	*Proof of Concept*	**WIM**	*Wide Area Infrastructure Manager*
IO	*Input-Output*	**PoP**	*Point of Presence*	**PRB**	*Physical Resource Block*

List of Figures

Figure 1. High-level organization of the Network Slice/Subnet design process according to the imperative provisioning style.....16
Figure 2. High-level organization of the Network Slice/Subnet design process according to the declarative provisioning style. 19
Figure 3. Landscape of organizations involved in shaping the Network Slicing technology and relationships between them. 23
Figure 4. The timeline of the author's research against the backdrop of Network Slicing technology development in the main industrial, standardization and open-source communities. 24
Figure 5. Requirements against the 4G, 5G and 6G networks and of the various Services they are expected to support. 27
Figure 6. Functional architecture of the 5G network divided into the main domains: User space, Radio Access Network (RAN), Core Network (CN), Transport Networks (TN) and Data Networks (DN).....30
Figure 7. Relationships between 5G NFs through numbered Reference Points, including examples of Service-based interactions between the AMF and SMF through the N11 Reference Point.....32
Figure 8. 5G network with several NSIs deployed, consisting of dedicated and shared NSSIs and NFs. Each NSI supports Services with different communication characteristics.....33
Figure 9. Basic telco networks management concepts standardized by 3GPP. 34
Figure 10. 3GPP Generic and 5G NRM management models corresponding to conceptual entities.....35
Figure 11. Correspondence between the concepts, generic management models and their concretizations within the 3GPP and ETSI NFV-MANO standards. 36
Figure 12. High-level overview of the ETSI NFV-MANO VNF Descriptor (VNFD) and Network Service Descriptor (NSD) structure. 38
Figure 13. ETSI Network Function Virtualization Management and Orchestration Framework (NFV-MANO).....41
Figure 14. NSI/NSSI lifecycle phases and the placement of the NSI/NSSI allocation process within them. 46
Figure 15. Examples of lifecycle relationships between the CSI, NSI and NSSI objects.....47
Figure 16. Relation between GST-NEST, ServiceProfile and SliceProfiles for different domains. 48
Figure 17. Runtime Network Slicing management models of 5G NRM and their mapping to the ETSI NFV-MANO runtime models. 50
Figure 18. Elements comprising the standardized Network Slicing orchestration stack.....55
Figure 19. Network Slicing orchestration architectural variants envisioned in the ONAP platform development. 61
Figure 20. The taxonomy of Slicing Entities according to six classification criteria: (1) lifecycle stage, (2) abstraction level, (3) concretization level, (4) class, (5) model and (6) type within the class. 68
Figure 21. Transitions between Design-time Slicing Entities of different concretization levels, which are performed during the design process: from an Abstract Template, through a Semi-concretized Descriptor until a fully-concretized Instance Descriptor. 70
Figure 22. Division of Slicing Entities into higher-level, lower-level and lowest-level. 71
Figure 23. Correspondence between Requirements attributes and Capabilities attributes in terms of name, type and semantics. 72
Figure 24. Confiuration Profiles containing sets of concretized values for the configurable attributes of a Slicing Entity and its internals. 74
Figure 25. Example of an MF Semi-concretized Descriptor concretized during a design process with configuration values for attributes required by its encapsulated entities and data structures. 79
Figure 26. Groups of configurations values to be included in the finalized design outcomes to fully configure the internal Slicing Entities for instantiation and operation in the designed Slice/Subnet.....82
Figure 27. Groups of information items to be conveyed by a standards-aligned Slice/Subnet Instance Descriptor.....84
Figure 28. Example of the Network Slice design outcomes produced as multiple Instance Descriptors that feed the orchestration blocks in a particular organization of the 3GPP-ETSI compliant Network Slicing orchestration stack. 85
Figure 29. High-level structure of 3GPP-compliant Network Slice/Subnet Instance Descriptors. 85
Figure 30. Graphical presentation of Requirement Decomposition Rules (RDR) transformations.....88
Figure 31. Graphical presentation of Capability Aggregation Rules (CAR) elements.....90
Figure 32. Schematic overview of the conceptual framework of a Network Slice/Subnet design process. 94
Figure 33. Roadmap of incremental Automation Scopes rollout in the Network Slice/Subnet preparation phase. 110
Figure 34. Levels of automation of a computer system, with the specification of human Operator involvement and autonomy border. 111
Figure 35. Overview of coarse-grained implementation-level phases of an end-to-end NSI/NSSI allocation workflow.....116
Figure 36. Architecture of the Subject Orchestrator before PoC implementation.....121
Figure 37. Capabilities attributes exposed by the MF Slicing Entities supported by the Subject Orchestrator.....124
Figure 38. Physical Resource Blocks (PRB) partitioning, including the explanation of the preferential and shared utilization thresholds. 124
Figure 39. Activities involved in the original NSSI allocation workflow in the Subject Orchestrator, before PoC implementation. 128
Figure 40. Aggregated Capabilities exposed by the NSSI objects supported by the Subject Orchestrator. 129
Figure 41. Subject Orchestrator's architecture after ASD PoC implementation.....131
Figure 42. UML diagram of the PoC elements incorporated inside of the Subject Orchestrator's components.....132
Figure 43. Slicing Entities abstract and concretized models supported in the PoC, as well as their relationship to SLS management models.....132
Figure 44. Concretizations of the abstract Deployment-time and Runtime Slicing Entities models supported in the PoC. 133
Figure 45. SLS Requirements and Capabilities management models.....134
Figure 46. Organization of Slicing Entities, SLS Requirements and Capabilities objects in a sample scenario, including graphical presentation of Requirements/Capabilities attributes transformation methods.....136
Figure 47. Models of Design-time Slicing Entities supported in the PoC, including models representing the abstract structure of an NSST and its Requirements Decomposition Rules (RDR). 137
Figure 48. Organization of the LOW_LEVEL_RAN_NSST, including exposed Capabilities, Abstract Nodes topology and RDRs between them. ...138
Figure 49. Models conveying results of selected design activities.....139
Figure 50. DesignPolicies model. 140
Figure 51. Functional modules containing implementation of selected design activities.....141
Figure 52. Organization of the end-to-end NSSI allocation workflow augmented with the ASD PoC.....144

Figure 53. Methods implemented in the ASD core functional module, encompassing logic of the main flows of the ASD workflow. 144
Figure 54. Organization of the EMBB_NSST "translation NSST" including the exposed Capabilities with high-level semantics and RDRs transforming high-level SLS Requirements into lower-level ones supported by the NSSI objects managed by the Subject Orchestrator. 146
Figure 55. Organization of the experimental testbed and input data feeding the Subject Orchestrator. 158
Figure 56. Geographical overview of the administered RAN domain located on the outskirts of a large city. 159
Figure 57. Inputs feeding the experimental testbed utilized for the functional evaluation of the PoC solution in a Case Study scenario. 160
Figure 58. Organization of gNB installations in the administered RAN domain - location of RUs and topology of Links between gNB MEs. . 161
Figure 59. Geographical coverage of the provisioned NSSIs and their assigned SliceProfiles after executing TC#1.sunny. 168
Figure 60. Geographical coverage of the provisioned NSSIs and their assigned SliceProfiles after executing TC#2.sunny. 173
Figure 61. Geographical coverage of the provisioned NSSIs and their assigned SliceProfiles after executing TC#3.sunny. 174
Figure 62. Geographical coverage of the provisioned NSSIs and their assigned SliceProfiles after executing TC#4.sunny. 177
Figure 63. Geographical coverage of the provisioned NSSIs and their assigned SliceProfiles after executing TC#5.sunny. 179
Figure 64. Inputs feeding the experimental testbed for the performance evaluation. 183
Figure 65. Classification of performance evaluation Test Cases. 186
Figure 66. Organization of Charts depicting evaluation results from performance Test Cases in various load conditions. 188
Figure 67. Timeline of 3GPP and ETSI standardization releases against the backdrop of 5G and 6G development schedules. 203

List of Charts

Chart 1. Performance results of executing Test Case A (allocation of new NSSI with 4 not loaded MFIs) in various load conditions. 189
Chart 2. Performance results of executing Test Case E (allocation of a new NSSI with 264 not loaded MFIs) in various load conditions. 190
Chart 3. Performance results of executing Test Case B (allocation of a new NSSI with 12 not loaded MFIs) in various load conditions. 191
Chart 4. Performance results of executing Test Case F (allocation of a new NSSI with 4 loaded MFIs) in various load conditions. 192
Chart 5. Performance results of executing Test Case J (allocation of a new NSSI with 264 loaded MFIs) in various load conditions. 192
Chart 6. Performance results of executing Test Case F (allocation of an existing NSSI with 132 not loaded MFIs) in various load conditions. 194
Chart 7. Performance results of executing Test Case S (allocation of an existing NSSI with 132 loaded MFIs) in various load conditions. 194

List of Sequence Diagrams

Sequence Diagram 1. Management steps needed to make a single MF, or a group of MFs (grouped into an NS), operational. 44
Sequence Diagram 2. Original flow of NSSI design and provisioning procedures in the Subject Orchestrator, before PoC implementation. . 126

List of Tables

Table 1. Meaning of the 3GPP-standardized Slice/Service Type (SST) values. 28
Table 2. Main functionalities and responsibilities of the standardized 5G RAN Network Functions' types. 30
Table 3. Main functionalities and responsibilities of the standardized 5G Core Network Functions' types in the particular layers. 31
Table 4. Semantics of Capabilities' values exposed by the Configuration Profiles of Slicing Entities with a given concretization level. 76
Table 5. Classification of config attributes to be supplied with concretized values in Slicing Entities of a given class to be fully concretized. . 81
Table 6. Comparison of RAN slicing mechanisms. 124
Table 7. Requirements Decomposition Rules (RDR) for high-level SLS Requirements against an eMBB-typed Subnet (SST=1). 146
Table 8. Requirements Decomposition Rules (RDR) for high-level SLS Requirements against an mIoT-typed Subnet (SST=3). 146
Table 9. Requirements Decomposition Rules (RDR) for high-level SLS Requirements against an V2X-typed Subnet (SST=4). 147
Table 10. Examples of lower-level RequirementsSets produced in the initial Requirements decomposition activity. 147
Table 11. DesignPolicies utilized in the Case Study. 162
Table 12. Initial state of MFI Slicing Entities' Capabilities and the aggregated Requirements posed against them. 163
Table 13. Case Study operational objects' tracking table, presenting the state of MFIs and NSSIs Slicing Entities' Capabilities and the aggregated Requirements assigned to them – initial state. 164
Table 14. Scenarios and Test Cases specification within the Case Study functional evaluation 165
Table 15. Case Study operational objects' tracking table – state after TC#1.sunny execution. 171
Table 16. Case Study operational objects' tracking table – state after TC#2.sunny execution. 174
Table 17. Case Study operational objects' tracking table – state after TC#3.sunny execution. 176
Table 18. Case Study operational objects' tracking table – state after TC#4.sunny execution. 178
Table 19. Case Study operational objects' tracking table – state after TC#5.sunny execution. 180
Table 20. Estimation of RAN domain geographical coverage assuming uniform distribution of 256 Cells in various configurations. 184
Table 21. Examples of Subject Orchestrator's operational load indicators considered in the performance evaluation. 185
Table 22. Specification of load conditions of the Subject Orchestrator. 185
Table 23. Specification of scale of NSSI-to-be-allocated in the performance evaluation Test Cases. 187

References

[1] NGMN Alliance, "Description of Network Slicing Concept," *Ngmn 5G P1*, vol. 1, no. September, p. 19, 2016, [Online]. Available: https://ngmn.org/wp-content/uploads/160113_NGMN_Network_Slicing_v1_0.pdf

[2] NGMN Alliance, "5G White Paper," 2015. [Online]. Available: https://www.ngmn.org/wp-content/uploads/NGMN_5G_White_Paper_V1_0.pdf

[3] A. J. Gonzalez *et al.*, "The Isolation Concept in the 5G Network Slicing," in *2020 European Conference on Networks and Communications (EuCNC)*, IEEE, Jun. 2020, pp. 12–16. doi: 10.1109/EuCNC48522.2020.9200939.

[4] P. Rost *et al.*, "Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 72–79, May 2017, doi: 10.1109/MCOM.2017.1600920.

[5] G. Wang, G. Feng, T. Q. S. Quek, S. Qin, R. Wen, and W. Tan, "Reconfiguration in Network Slicing—Optimizing the Profit and Performance," *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 2, pp. 591–605, Jun. 2019, doi: 10.1109/TNSM.2019.2899609.

[6] F. Wei, G. Feng, Y. Sun, Y. Wang, S. Qin, and Y.-C. Liang, "Network Slice Reconfiguration by Exploiting Deep Reinforcement Learning With Large Action Space," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 4, pp. 2197–2211, Dec. 2020, doi: 10.1109/TNSM.2020.3019248.

[7] F. Wei, S. Qin, G. Feng, Y. Sun, J. Wang, and Y.-C. Liang, "Hybrid Model-Data Driven Network Slice Reconfiguration by Exploiting Prediction Interval and Robust Optimization," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 2, pp. 1426–1441, Jun. 2022, doi: 10.1109/TNSM.2021.3138560.

[8] X. Zhou, R. Li, T. Chen, and H. Zhang, "Network slicing as a service: enabling enterprises' own software-defined cellular networks," *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 146–153, Jul. 2016, doi: 10.1109/MCOM.2016.7509393.

[9] 3GPP, "Telecommunication management; Study on management and orchestration of network slicing for next generation network, TR 28.801, V15.1.0 (2018-01)." [Online]. Available: https://www.3gpp.org/ftp//Specs/archive/28_series/28.801/28801-f10.zip

[10] 3GPP, "Technical Specification Group Services and System Aspects; Telecommunication management; Study on management aspects of communication services, TR 28.805 V16.1.0 (2019-12)", [Online]. Available: https://www.3gpp.org/ftp//Specs/archive/28_series/28.805/28805-g10.zip

[11] 3GPP, "Management and orchestration; Concepts, use cases and requirements, TS 28.530, V19.0.0 (2025-03)", [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/28_series/28.530/28530-j00.zip

[12] A. Çubukçu, A. Kavak, Ö. Çubukçu, Y. S. Sandal, and K. Küçük, "A Study on Slice-Aware Industrial Edge Applications for Next-Generation Private Networks," in *2023 4th International Informatics and Software Engineering Conference (IISEC)*, IEEE, Dec. 2023, pp. 1–5. doi: 10.1109/IISEC59749.2023.10391053.

[13] S. Wijethilaka and M. Liyanage, "A Novel Network Slicing based Security-as-a-Service (SECaaS) Framework for Private 5G Networks," in *2022 IEEE Latin-American Conference on Communications (LATINCOM)*, IEEE, Nov. 2022, pp. 1–6. doi: 10.1109/LATINCOM56090.2022.10000486.

[14] X. You *et al.*, "Towards 6G wireless communication networks: vision, enabling technologies, and new paradigm shifts," *Sci. China Inf. Sci.*, vol. 64, no. 1, p. 110301, Jan. 2021, doi: 10.1007/s11432-020-2955-6.

[15] J. T. J. Penttinen, "On 6G Visions and Requirements," *J. ICT Stand.*, Dec. 2021, doi: 10.13052/jicts2245-800X.931.

[16] A. Dogra, R. K. Jha, and S. Jain, "A Survey on Beyond 5G Network With the Advent of 6G: Architecture and Emerging Technologies," *IEEE Access*, vol. 9, pp. 67512–67547, 2021, doi: 10.1109/ACCESS.2020.3031234.

[17] A. Fernandez-Fernandez, E. Coronado, A. Erspamer, G. Samaras, V. Theodorou, and S. Siddiqui, "Unlocking the Path Towards Intelligent Telecom Marketplaces for Beyond 5G and 6G Networks," *IEEE Commun. Mag.*, pp. 1–7, 2023, doi: 10.1109/MCOM.002.2200266.

[18] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis, and X. Costa-Perez, "Optimising 5G infrastructure markets: The business of network slicing," *Proc. - IEEE INFOCOM*, 2017, doi: 10.1109/INFOCOM.2017.8057045.

[19] S. Kukliński, L. Tomaszewski, K. Kozłowski, and S. Pietrzyk, "Business models of network slicing," *Proc. 2018 9th Int. Conf. Netw. Futur. NOF 2018*, pp. 39–43, 2018, doi: 10.1109/NOF.2018.8597858.

[20] M. Vincenzi, E. Lopez-Aguilera, and E. Garcia-Villegas, "Maximizing Infrastructure Providers' Revenue Through Network Slicing in 5G," *IEEE Access*, vol. 7, pp. 128283–128297, 2019, doi: 10.1109/ACCESS.2019.2939935.

[21] TM Forum, "Network Slice Monetization v1.0.0 (IG1179A)," 2021. [Online]. Available: https://www.tmforum.org/resources/standard/ig1179a-network-slice-monetization-v1-0-0/

[22] ABI Research, "5G Slicing Revenue to Grow from US$309 Million in 2022 to Approximately US$24 Billion in 2028." [Online]. Available: https://www.abiresearch.com/press/5g-slicing-revenue-to-grow-from-us309-million-in-2022-to-approximately-us24-billion-in-2028/

[23] D. Meyer, "Telefónica, Ericsson, Google Show Network Slicing Progress," 2022. [Online]. Available: https://www.sdxcentral.com/articles/news/telefonica-ericsson-google-show-network-slicing-progress/2022/07/

[24] G. Brown, "When to launch 5G network slicing," 2022. https://www.lightreading.com/when-to-launch-5g-network-slicing/a/d-id/782207 (accessed April 15, 2025).

[25] Heavy Reading, "5G Network Slicing Operator Survey," 2022. [Online]. Available: https://mrkt-lp.flywheelsites.com/heavy-reading-5g-network-slicing/

[26] O. G. Granados, "Apple Brings 5G Network Slicing Closer To Reality," 2023. https://www.forrester.com/blogs/apple-brings-5g-network-slicing-closer-to-reality/

[27] L. Hardesty, "Deutsche Telekom offers a commercial 5G network slicing service," 2023. https://www.fierce-network.com/tech/deutsche-telekom-offers-commercial-5g-network-slicing-service

[28] J. Ordonez-Lucena, P. Ameigeiras, L. M. Contreras, J. Folgueira, and D. R. López, "On the Rollout of Network Slicing in Carrier Networks: A Technology Radar," *Sensors*, vol. 21, no. 23, p. 8094, Dec. 2021, doi: 10.3390/s21238094.

[29] GSM Association, "E2E Network Slicing Architecture - White Paper, Version 2.0, 2022-05," 2022. [Online]. Available:

https://www.gsma.com/newsroom/wp-content/uploads//NG.127-v2.0.pdf
[30] P. Wyszkowski, J. Kienig, K. Zieliński, Ł. Czekierda, and M. Zawadzki, "Comprehensive Tutorial on the Organization of a Standards-Aligned Network Slice/Subnet Design Process and Opportunities for Its Automation," *IEEE Commun. Surv. Tutorials*, vol. 26, no. 2, pp. 1386–1445, 2024, doi: 10.1109/COMST.2023.3341249.
[31] 3GPP, "Management and orchestration; Provisioning; TS 28.531 V19.0.0 (2025-01)." [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/28_series/28.531/28531-j00.zip
[32] 3GPP, "Management and orchestration; 5G Network Resource Model (NRM); Stage 2 and stage 3, TS 28.541, V19.3.0 (2025-03)," 2025. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/28_series/28.541/28541-j30.zip
[33] "TELUS and Ericsson accelerating commercial 5G applications with standards-based network slicing orchestration." https://www.newswire.ca/news-releases/telus-and-ericsson-accelerating-commercial-5g-applications-with-standards-based-network-slicing-orchestration-859511406.html (accessed April 15, 2025).
[34] "Ericsson, Deutsche Telekom and Samsung complete 'world-first' 5G E2E network slicing trial – 5G Observatory." https://5gobservatory.eu/ericsson-deutsche-telekom-and-samsung-complete-word-first-5g-e2e-network-slicing-trial/ (accessed April 15, 2025).
[35] GSM Association, "Generic Network Slice Template, NG.116 V8.0 (2023-01)." [Online]. Available: https://www.gsma.com/newsroom/wp-content/uploads/NG.116-v8.0-1.pdf
[36] P. M. Mell and T. Grance, "The NIST definition of cloud computing," Gaithersburg, MD, 2011. doi: 10.6028/NIST.SP.800-145.
[37] M. Lin and Y. Zhao, "Artificial intelligence-empowered resource management for future wireless communications: A survey," *China Commun.*, vol. 17, no. 3, pp. 58–77, Mar. 2020, doi: 10.23919/JCC.2020.03.006.
[38] J. Martin-Perez, F. Malandrino, C. F. Chiasserini, M. Groshev, and C. J. Bernardos, "KPI Guarantees in Network Slicing," *IEEE/ACM Trans. Netw.*, vol. 30, no. 2, pp. 655–668, Apr. 2022, doi: 10.1109/TNET.2021.3120318.
[39] A. Papageorgiou, A. Fernandez-Fernandez, L. Ochoa-Aday, M. S. Pelaez, and M. Shuaib Siddiqui, "SLA Management Procedures in 5G Slicing-based Systems," in *2020 European Conference on Networks and Communications (EuCNC)*, IEEE, 2020, pp. 7–11. doi: 10.1109/EuCNC48522.2020.9200904.
[40] H. Wang, Y. Wu, G. Min, J. Xu, and P. Tang, "Data-driven dynamic resource scheduling for network slicing: A Deep reinforcement learning approach," *Inf. Sci. (Ny).*, vol. 498, pp. 106–116, Sep. 2019, doi: 10.1016/j.ins.2019.05.012.
[41] D. Ferreira, A. Braga Reis, C. Senna, and S. Sargento, "A Forecasting Approach to Improve Control and Management for 5G Networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 2, pp. 1817–1831, Jun. 2021, doi: 10.1109/TNSM.2021.3056222.
[42] V. P. Kafle, P. Martinez-Julia, and T. Miyazawa, "Automation of 5G Network Slice Control Functions with Machine Learning," *IEEE Commun. Stand. Mag.*, vol. 3, no. 3, pp. 54–62, Sep. 2019, doi: 10.1109/MCOMSTD.001.1900010.
[43] J. A. Ordonez-Lucena, "Management and Orchestration of Network Slicing in Public-Private 5G Networks," University of Granada, 2022. [Online]. Available: https://produccioncientifica.ugr.es/documentos/643841d6cd47bb6af696c570
[44] J. Kienig and P. Wyszkowski, "Method and apparatus for transmitting/receiving network slice configuration in communication," Patent No. KR102761449B1. Daejeon, Korean Intellectual Property Office, 2021
[45] P. Wyszkowski, J. Kienig, D. Ho, G. Chandrasekaran, Ł. Filimowski, and M. Pilarczyk, "Apparatus and method for generating Network Slice in wireless communication system," Patent Application No. US20220353138A1. Washington, DC: U.S. Patent and Trademark Office, 2020
[46] P. Wyszkowski, J. Kienig, D. Ho, G. Chandrasekaran, Ł. Filimowski, and M. Pilarczyk, "Apparatus and method for generating Network Slice in wireless communication system," Patent Application No. KR20210101373A. Daejeon, Korean Intellectual Property Office, 2020
[47] J. Kienig, P. Wyszkowski, and M. Zawadzki, "Management node in wireless communication system having Network Slicing environment, and operating method thereof," Patent Application No. US2024356802A1. Washington, DC: U.S. Patent and Trademark Office, 2023
[48] P. Wyszkowski, J. Kienig, M. Lisowski, and M. Zawadzki, "Management node in wireless communication system having Network Slicing environment, and operating method thereof," Patent Application No. US20240381178A1. Washington, DC: U.S. Patent and Trademark Office, 2023
[49] P. Wyszkowski, J. Kienig, M. Lisowski, and M. Zawadzki, "Management node in wireless communication system having Network Slicing environment, and operating method thereof," Patent Application No. KR20240163489A. Daejeon, Korean Intellectual Property Office, 2023
[50] D. Ho, J. Kienig, P. Wyszkowski, G. Chandrasekaran, and I. Yun, "Method and device for providing Network Slice," Patent Application No. US2023045460A1. Washington, DC: U.S. Patent and Trademark Office, 2020
[51] D. Ho, J. Kienig, P. Wyszkowski, G. Chandrasekaran, and I. Yun, "Method and device for providing Network Slice," Patent Application No. KR20210127566A. Daejeon, Korean Intellectual Property Office, 2020
[52] J. Kienig, P. Wyszkowski, and R. Gibas, "Device performing communication and method for operating thereof," Patent Application No. KR20220142720A. Daejeon, Korean Intellectual Property Office, 2021
[53] J. Kienig, P. Wyszkowski, and R. Gibas, "Device performing communication and method for operating thereof," Patent Application No. US20223379911A1. Washington, DC: U.S. Patent and Trademark Office, 2021
[54] J. Kienig, P. Wyszkowski, and R. Gibas, "Device for establishing communication, and operation method thereof," Patent Application No. EP4271047A1. Munich, Germany. European Patent Office, 2021
[55] G. Grzesiak *et al.*, "Method and apparatus for reducing latency of streaming service by Network Slices parallel in wireless communication system," Patent Application No. US2023336605A1. Washington, DC: U.S. Patent and Trademark Office, 2022
[56] G. Grzesiak *et al.*, "Method and apparatus for reducing latency of streaming service by Network Slices parallel in wireless communication system," Patent Application No. KR20230142987A. Daejeon, Korean Intellectual Property Office, 2022
[57] J. Kienig, P. Wyszkowski, and M. Zawadzki, "Management node in wireless communication system having Network Slicing environment, and operating method thereof," Patent Application No. KR20240156906A. Daejeon, Korean Intellectual Property Office, 2023
[58] 3GPP, "Management and orchestration; Provisioning; TS 28.531 V1.3.0 (2018-08)," 2018. [Online]. Available: https://www.3gpp.org/ftp//Specs/archive/28_series/28.531/28531-130.zip
[59] 3GPP, "Management and orchestration of 5G networks; Network Resource Model (NRM); Stage 2 and stage 3; TS 28.541 V1.1.0 (2018-07)," 2018. [Online]. Available: https://www.3gpp.org/ftp//Specs/archive/28_series/28.541/28541-110.zip

[60] R. Rokui, H. Shunsuke, K. Makhijani, L. M. Contreras, and J. Tantsura, "IETF Definition of Transport Slice," 2020. [Online]. Available: https://datatracker.ietf.org/doc/draft-nsdt-teas-transport-slice-definition/

[61] R. Rokui *et al.*, "5G Transport Slice Connectivity Interface," Internet Engineering Task Force, 2020. [Online]. Available: https://datatracker.ietf.org/doc/draft-rokui-5g-transport-slice/

[62] S. Barguil, L. M. Contreras, V. Lopez, O. G. de Dios, and M. Boucadair, "Applicability of IETF-Defined Service and Network Data Models for Network Slice Service Management," 2021. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-teas-ns-models-applicability/

[63] L. M. Contreras, R. Rokui, J. Tantsura, B. Wu, and X. Liu, "IETF Network Slice Controller and its Associated Data Models," Internet Engineering Task Force, 2020. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-teas-ns-controller-models/03/

[64] ITU-R, "Recommendation ITU-R M.2150-2 (12/2023); Detailed specifications of the terrestrial radio interfaces of International Mobile Telecommunications-2020 (IMT-2020)," 2023. [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.2150-2-202312-I!!PDF-E.pdf

[65] ITU-R, "Recommendation ITU-R M.1457-15 (10/2020); Detailed specifications of the terrestrial radio interfaces of International Mobile Telecommunications-2000 (IMT-2000)," 2020. [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.1457-15-202010-I!!PDF-E.pdf

[66] ITU-R, "Recommendation ITU-R M.2012-6 (12/2023); Detailed specifications of the terrestrial radio interfaces of International Mobile Telecommunications-Advanced (IMT-Advanced)," 2023. [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.2012-6-202312-I!!PDF-E.pdf

[67] ITU-R, "Recommendation ITU-R M.2160-0 (11/2023); Framework and overall objectives of the future development of IMT for 2030 and beyond," 2023. [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.2160-0-202311-I%21%21PDF-E.pdf

[68] 3GPP, "System architecture for the 5G System (5GS), TS 23.501, V19.3.0 (2025-03)." [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.501/23501-j30.zip

[69] 3GPP, "Service requirements for the 5G system; Stage 1, TS 22.261 V20.2.0 (2025-03)," 2025. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/22_series/22.261/22261-k20.zip

[70] J. Lewis and M. Fowler, "Microservices," 2014. https://martinfowler.com/articles/microservices.html (accessed April 15, 2025).

[71] 3GPP, "Management and orchestration; Architecture framework, TS 28.533, V19.1.0 (2025-03)," 2025. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/28_series/28.533/28533-j10.zip

[72] 3GPP, "Procedures for the 5G System (5GS); Stage 2, TS 23.502, V19.3.0 (2023-06)," 2025. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.502/23502-j30.zip

[73] 3GPP, "Management and orchestration; Generic management services, TS 28.532, V19.0.0 (2025-03)," 2025. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/28_series/28.532/28532-j00.zip

[74] B. Jedari, G. Premsankar, G. Illahi, M. Di Francesco, A. Mehrabi, and A. Yla-Jaaski, "Video Caching, Analytics, and Delivery at the Wireless Edge: A Survey and Future Directions," *IEEE Commun. Surv. Tutorials*, vol. 23, no. 1, pp. 431–471, 2021, doi: 10.1109/COMST.2020.3035427.

[75] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A Survey on Mobile Augmented Reality With 5G Mobile Edge Computing: Architectures, Applications, and Technical Aspects," *IEEE Commun. Surv. Tutorials*, vol. 23, no. 2, pp. 1160–1192, 2021, doi: 10.1109/COMST.2021.3061981.

[76] China Mobile, "C-RAN: the road towards green RAN," *China Mob. White Pap. ver 2.5*, vol. 5, pp. 15–16, 2011, [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:C-RAN+The+Road+Towards+Green+RAN#0

[77] R. Cassan, "How to Make Automated 5G Network Slicing Work," 2019. https://www.spirent.com/blogs/the-5g-transformation-needs-automated-assurance-now-2-network-slice-activation (accessed April 15, 2025).

[78] 3GPP, "Telecommunication management; Principles and high level requirements, TS 32.101, V18.0.0 (2024-04)," 2024. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/32_series/32.101/32101-i00.zip

[79] 3GPP, "Vocabulary for 3GPP Specifications, TR 21.905, V18.0.0 (2024-04)," 2024. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/21_series/21.905/21905-i00.zip

[80] 3GPP, "Telecommunication management; Generic Network Resource Model (NRM) Integration Reference Point (IRP); Information Service (IS), TS 28.622, V19.3.0 (2025-03)." [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/28_series/28.622/28622-j30.zip

[81] ETSI, "Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; VNF Descriptor and Packaging Specification, GS NFV-IFA 011 V5.2.1 (2024-12)," 2025. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/011/05.02.01_60/gs_NFV-IFA011v050201p.pdf

[82] ETSI, "Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Interface and Information Model Specification for Multi-Site Connectivity Services, GS NFV-IFA 032 V5.2.1 (2024-11)." [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/032/05.02.01_60/gs_NFV-IFA032v050201p.pdf

[83] ETSI, "Network Functions Virtualisation (NFV) Release 3;Protocols and Data Models; Report on protocol and data model solutions for Multi-site Connectivity Services; GR NFV-SOL 017 V3.3.1 (2021-05)." [Online]. Available: https://www.etsi.org/deliver/etsi_gr/NFV-SOL/001_099/017/03.03.01_60/gr_NFV-SOL017v030301p.pdf

[84] ETSI, "Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Network Service Templates Specification, GS NFV-IFA 014 V5.2.1 (2024-11)", [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/014/05.02.01_60/gs_NFV-IFA014v050201p.pdf

[85] ETSI, "Network Functions Virtualisation (NFV) Release 5; Protocols and Data Models; NFV descriptors based on TOSCA specification, GS NFV-SOL 001 V5.2.1 (2024-12)." [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/001/05.02.01_60/gs_NFV-SOL001v050201p.pdf

[86] ETSI, "Network Functions Virtualisation (NFV) Release 5; Protocols and Data Models; VNF Package and PNFD Archive specification, GS NFV-SOL 004 V5.1.1 (2024-07)." [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/004/05.01.01_60/gs_NFV-SOL004v050101p.pdf

[87] J. Ordonez-Lucena *et al.*, "The Creation Phase in Network Slicing: From a Service Order to an Operative Network Slice," *2018 Eur. Conf. Networks Commun. EuCNC 2018*, pp. 31–36, 2018, doi: 10.1109/EuCNC.2018.8443255.

[88] ETSI, "Network Functions Virtualisation (NFV) Release 5; Protocols and Data Models; RESTful protocols specification for the Or-Vnfm Reference Point, GS NFV-SOL 003 V5.2.1 (2025-01)." [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-

SOL/001_099/003/05.02.01_60/gs_NFV-SOL003v050201p.pdf
[89] ETSI, "Network Functions Virtualisation (NFV) Release 5; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point, GS NFV-SOL 005 V5.2.1 (2024-12)", [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/005/05.02.01_60/gs_NFV-SOL005v050201p.pdf
[90] ETSI, "Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Or-Vnfm reference point - Interface and Information Model Specification, GS NFV-IFA 007 V5.2.1 (2024-12)", [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/007/05.02.01_60/gs_NFV-IFA007v050201p.pdf
[91] M. Björklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," no. 6020. in Request for Comments. RFC Editor, 2010. doi: 10.17487/RFC6020.
[92] ETSI, "Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Vi-Vnfm reference point - Interface and Information Model Specification, GS NFV-IFA 006 V5.2.1 (2024-11)." [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/006/05.02.01_60/gs_NFV-IFA006v050201p.pdf
[93] ETSI, "Network Functions Virtualisation (NFV) Release 5; Protocols and Data Models; RESTful protocols specification for the Ve-Vnfm Reference Point, GS NFV-SOL 002 V5.2.1 (2024-12)." [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/002/05.02.01_60/gs_NFV-SOL002v050201p.pdf
[94] ETSI, "Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Ve-Vnfm reference point - Interface and Information Model Specification, GS NFV-IFA 008 V5.2.1 (2024-12)." [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/008/05.02.01_60/gs_NFV-IFA008v050201p.pdf
[95] 3GPP, "Management and orchestration; Provisioning, TS 28.531 V16.2.0 (2019-06)", [Online]. Available: https://www.3gpp.org/ftp//Specs/archive/28_series/28.531/28531-g20.zip
[96] Affirmed Networks, "4G & 5G-Ready Network Slicing," 2017. https://www.affirmednetworks.com/pdfs/4g-5g-ready-network-slicing.pdf/ (accessed April 15, 2025).
[97] A. Galis and S. Clayman, "NECOS System Architecture and Plarform Specification V2," 2019. [Online]. Available: http://www.maps.upc.edu/public/necos_d3.2.v4.11_final_web.pdf
[98] H. O. Otieno, B. Malila, and J. Mwangama, "Deployment and Management of Intelligent End-to-End Network Slicing in 5G and Beyond 5G Networks: A Systematic Review," *IEEE Access*, vol. 12, pp. 190411–190433, 2024, doi: 10.1109/ACCESS.2024.3516051.
[99] M. Chahbar, G. Diaz, A. Dandoush, C. Cerin, and K. Ghoumid, "A Comprehensive Survey on the E2E 5G Network Slicing Model," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 1, pp. 49–62, Mar. 2021, doi: 10.1109/TNSM.2020.3044626.
[100] V. Q. Rodriguez, F. Guillemin, and A. Boubendir, "Automating the deployment of 5G Network Slices using ONAP," in *2019 10th International Conference on Networks of the Future (NoF)*, IEEE, Oct. 2019, pp. 32–39. doi: 10.1109/NoF47743.2019.9015043.
[101] Open Source MANO, "OSM NST Information Model." https://osm.etsi.org/wikipub/index.php/OSM_Information_Model#NST_tree (accessed April 15, 2025).
[102] 3GPP, "Management and orchestration; Provisioning, TS 28.531 V16.3.0 (2019-09)." [Online]. Available: https://www.3gpp.org/ftp//Specs/archive/28_series/28.531/28531-g30.zip
[103] X. Li, R. Ni, J. Chen, Y. Lyu, Z. Rong, and R. Du, "End-to-End Network Slicing in Radio Access Network, Transport Network and Core Network Domains," *IEEE Access*, vol. 8, pp. 29525–29537, 2020, doi: 10.1109/ACCESS.2020.2972105.
[104] I. Afolabi, M. Bagaa, W. Boumezer, and T. Taleb, "Toward a Real Deployment of Network Services Orchestration and Configuration Convergence Framework for 5G Network Slices," *IEEE Netw.*, vol. 35, no. 1, pp. 242–250, Jan. 2021, doi: 10.1109/MNET.011.2000146.
[105] S. Clayman *et al.*, "The NECOS Approach to End-to-End Cloud-Network Slicing as a Service," *IEEE Commun. Mag.*, vol. 59, no. 3, pp. 91–97, Mar. 2021, doi: 10.1109/MCOM.001.2000702.
[106] C. Zhang *et al.*, "Architectural Design and Life Cycle Management of Network Slicing for Software-Defined Optical Access Networks," *IEEE Netw.*, vol. 36, no. 2, pp. 82–88, Mar. 2022, doi: 10.1109/MNET.001.2100508.
[107] I. Afolabi, J. Prados-Garzon, M. Bagaa, T. Taleb, and P. Ameigeiras, "Dynamic Resource Provisioning of a Scalable E2E Network Slicing Orchestration System," *IEEE Trans. Mob. Comput.*, vol. 19, no. 11, pp. 2594–2608, Nov. 2020, doi: 10.1109/TMC.2019.2930059.
[108] A. Cardenas, D. Fernandez, C. M. Lentisco, R. F. Moyano, and L. Bellido, "Enhancing a 5G Network Slicing Management Model to Improve the Support of Mobile Virtual Network Operators," *IEEE Access*, vol. 9, pp. 131382–131399, 2021, doi: 10.1109/ACCESS.2021.3114645.
[109] P. Alemany *et al.*, "A KPI-Enabled NFV MANO Architecture for Network Slicing with QoS," *IEEE Commun. Mag.*, vol. 59, no. 7, pp. 44–50, Jul. 2021, doi: 10.1109/MCOM.001.2001077.
[110] M. Shariat *et al.*, "A Flexible Network Architecture for 5G Systems," *Wirel. Commun. Mob. Comput.*, vol. 2019, pp. 1–19, Feb. 2019, doi: 10.1155/2019/5264012.
[111] M. A. Habibi, B. Han, F. Z. Yousaf, and H. D. Schotten, "How Should Network Slice Instances Be Provided to Multiple Use Cases of a Single Vertical Industry?," *IEEE Commun. Stand. Mag.*, vol. 4, no. 3, pp. 53–61, Sep. 2020, doi: 10.1109/MCOMSTD.001.1900050.
[112] K. Abbas, M. Afaq, T. A. Khan, A. Mehmood, and W.-C. Song, "IBNSlicing: Intent-Based Network Slicing Framework for 5G Networks using Deep Learning," in *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, IEEE, Sep. 2020, pp. 19–24. doi: 10.23919/APNOMS50412.2020.9237008.
[113] S. Kukliński and L. Tomaszewski, "DASMO: A scalable approach to network slices management and orchestration," in *2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–6.
[114] T. A. Khan, K. Abbass, A. Rafique, A. Muhammad, and W.-C. Song, "Generic Intent-based Networking Platform for E2E Network Slice Orchestration & Lifecycle Management," in *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, IEEE, Sep. 2020, pp. 49–54. doi: 10.23919/APNOMS50412.2020.9236962.
[115] S. Kukliński *et al.*, "A reference architecture for network slicing," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, IEEE, 2018, pp. 217–221. doi: 10.1109/NETSOFT.2018.8460057.
[116] T. Taleb, I. Afolabi, K. Samdanis, and F. Z. Yousaf, "On Multi-Domain Network Slicing Orchestration Architecture and Federated Resource Control," *IEEE Netw.*, vol. 33, no. 5, pp. 242–252, Sep. 2019, doi: 10.1109/MNET.2018.1800267.
[117] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," *Comput. Networks*, vol. 167, p. 106984, 2020, doi: 10.1016/j.comnet.2019.106984.
[118] A. Boubendir *et al.*, "Federation of Cross-Domain Edge Resources: A Brokering Architecture for Network Slicing," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, IEEE, Jun. 2018, pp. 415–423. doi: 10.1109/NETSOFT.2018.8460114.
[119] J. Ordonez-Lucena, C. Tranoris, and J. Rodrigues, "Modeling Network Slice as a Service in a Multi-Vendor 5G Experimentation

Ecosystem," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, Jun. 2020, pp. 1–6. doi: 10.1109/ICCWorkshops49005.2020.9145225.
[120] S. Kukliński, L. Tomaszewski, R. Kołakowski, and P. Chemouil, "6G-LEGO: A framework for 6G network slices," *J. Commun. Networks*, vol. 23, no. 6, pp. 442–453, Dec. 2021, doi: 10.23919/JCN.2021.000025.
[121] K. Abbas, T. A. Khan, M. Afaq, and W.-C. Song, "Network Slice Lifecycle Management for 5G Mobile Networks: An Intent-Based Networking Approach," *IEEE Access*, vol. 9, pp. 80128–80146, 2021, doi: 10.1109/ACCESS.2021.3084834.
[122] C. Parada *et al.*, "5Gtango: A Beyond-Mano Service Platform," in *2018 European Conference on Networks and Communications (EuCNC)*, 2018, pp. 26–30.
[123] "OSM Scope and Functionality." https://osm.etsi.org/wikipub/index.php/OSM_Scope_and_Functionality (accessed April 15, 2025).
[124] A. Ishaq *et al.*, "Service-Based Management Architecture for On-Demand Creation, Configuration, and Control of a Network Slice Subnet," in *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*, IEEE, Jun. 2022, pp. 275–277. doi: 10.1109/NetSoft54395.2022.9844081.
[125] TM Forum, "TMF641 Service Ordering API User Guide v4.1.0." [Online]. Available: https://projects.tmforum.org/wiki/display/API/TMF641+Service+Ordering+API+User+Guide+v4.1.0
[126] TM Forum, "TMF638 Service Inventory API REST Specification R18.5.0", [Online]. Available: https://projects.tmforum.org/wiki/display/API/TMF638+Service+Inventory+API+REST+Specification+R18.5.0
[127] 5GPagoda, "A network slice for every service | 5GPagoda Project," 2019. https://5g-pagoda.aalto.fi/ (accessed April 15, 2025).
[128] Y. Wu, H.-N. Dai, H. Wang, Z. Xiong, and S. Guo, "A Survey of Intelligent Network Slicing Management for Industrial IoT: Integrated Approaches for Smart Transportation, Smart Energy, and Smart Factory," *IEEE Commun. Surv. Tutorials*, vol. 24, no. 2, pp. 1175–1211, 2022, doi: 10.1109/COMST.2022.3158270.
[129] S. Wijethilaka and M. Liyanage, "Survey on Network Slicing for Internet of Things Realization in 5G Networks," *IEEE Commun. Surv. Tutorials*, vol. 23, no. 2, pp. 957–994, 2021, doi: 10.1109/COMST.2021.3067807.
[130] A. Papageorgiou, A. Fernández-Fernández, S. Siddiqui, and G. Carrozzo, "On 5G network slice modelling: Service-, resource-, or deployment-driven?," *Comput. Commun.*, vol. 149, pp. 232–240, Jan. 2020, doi: 10.1016/j.comcom.2019.10.024.
[131] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions," *IEEE Commun. Surv. Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018, doi: 10.1109/COMST.2018.2815638.
[132] C. Casetti *et al.*, "Network slices for vertical industries," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, IEEE, Apr. 2018, pp. 254–259. doi: 10.1109/WCNCW.2018.8368981.
[133] M. Wang, B. Cheng, and J. Chen, "A Lightweight SOA-Based Network Slicing Creation System," in *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*, IEEE, Oct. 2020, pp. 389–396. doi: 10.1109/CLOUD49709.2020.00060.
[134] J. F. Santos *et al.*, "Breaking Down Network Slicing: Hierarchical Orchestration of End-to-End Networks," *IEEE Commun. Mag.*, vol. 58, no. 10, pp. 16–22, Oct. 2020, doi: 10.1109/MCOM.001.2000406.
[135] Q.-T. Luu, S. Kerboeuf, and M. Kieffer, "Uncertainty-Aware Resource Provisioning for Network Slicing," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 1, pp. 79–93, Mar. 2021, doi: 10.1109/TNSM.2021.3058139.
[136] K. Abbas, M. Afaq, T. Ahmed Khan, A. Rafiq, and W.-C. Song, "Slicing the Core Network and Radio Access Network Domains through Intent-Based Networking for 5G Networks," *Electronics*, vol. 9, no. 10, p. 1710, Oct. 2020, doi: 10.3390/electronics9101710.
[137] A. Rafiq, A. Mehmood, T. Ahmed Khan, K. Abbas, M. Afaq, and W.-C. Song, "Intent-Based End-to-End Network Service Orchestration System for Multi-Platforms," *Sustainability*, vol. 12, no. 7, p. 2782, Apr. 2020, doi: 10.3390/su12072782.
[138] T. A. Khan, A. Mehmood, J. J. Diaz Ravera, A. Muhammad, K. Abbas, and W.-C. Song, "Intent-Based Orchestration of Network Slices and Resource Assurance using Machine Learning," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, IEEE, Apr. 2020, pp. 1–2. doi: 10.1109/NOMS47738.2020.9110408.
[139] N. Gritli, F. Khendek, and M. Toeroe, "Network Slice Provisioning Taking Into Account Tenant Intents and Operator Policies," in *2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, IEEE, Nov. 2021, pp. 167–173. doi: 10.1109/NFV-SDN53031.2021.9665134.
[140] B. Martini, M. Gharbaoui, and P. Castoldi, "Intent-based network slicing for SDN vertical services with assurance: Context, design and preliminary experiments," *Futur. Gener. Comput. Syst.*, vol. 142, pp. 101–116, May 2023, doi: 10.1016/j.future.2022.12.033.
[141] P. Wyszkowski, "How to achieve automated, standards-aligned Network Slices design and allocation," 2024. https://research.samsung.com/blog/How-to-achieve-automated-standards-aligned-Network-Slices-design-and-allocation
[142] ONAP, "E2E Network Slicing Use Case." https://docs.onap.org/projects/onap-integration/en/latest/docs_E2E_network_slicing.html (accessed April 15, 2025).
[143] ONAP, "ONAP 'Oslo' release architecture and orchestration architecture variants," 2025. https://docs.onap.org/en/oslo/ecosystem/architecture/index.html#orchestration (accessed April 15, 2025).
[144] A. Clemm, L. Ciavaglia, L. Z. Granville, and J. Tantsura, "Intent-Based Networking - Concepts and Definitions," Oct. 2022. doi: 10.17487/RFC9315.
[145] A. Leivadeas and M. Falkner, "A Survey on Intent Based Networking," *IEEE Commun. Surv. Tutorials*, pp. 1–1, 2022, doi: 10.1109/COMST.2022.3215919.
[146] 3GPP, "Study on scenarios for Intent driven management services for mobile networks, TR 28.812 V17.1.0 (2020-12)." [Online]. Available: https://www.3gpp.org/ftp//Specs/archive/28_series/28.812/28812-h10.zip
[147] 3GPP, "Management and orchestration; Intent driven management services for mobile networks, TS 28.312 V19.1.0 (2025-03)," 2025. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/28_series/28.312/28312-j10.zip
[148] A. S. Jacobs, R. J. Pfitscher, R. A. Ferreira, and L. Z. Granville, "Refining Network Intents for Self-Driving Networks," in *Proceedings of the Afternoon Workshop on Self-Driving Networks*, New York, NY, USA: ACM, Aug. 2018, pp. 15–21. doi: 10.1145/3229584.3229590.
[149] J. McNamara, E. Aumayr, L. Fallon, and E. Fallon, "A Flexible Interpreter For Intent Realisation," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, IEEE, Apr. 2022, pp. 1–6. doi: 10.1109/NOMS54207.2022.9789910.
[150] Y. Drif *et al.*, "An extensible network slicing framework for satellite integration into 5G," *Int. J. Satell. Commun. Netw.*, vol. 39, no. 4, pp. 339–357, Jul. 2021, doi: 10.1002/sat.1387.
[151] K. Govindarajan *et al.*, "Closed loop optimization of 5G network slices," in *Proceedings of the 23rd International Middleware Conference Industrial Track*, New York, NY, USA: ACM, Nov. 2022, pp. 29–35. doi: 10.1145/3564695.3564776.
[152] G. Bhatnagar, S. Gupta, and M. Verma, "Adaptive Conditional Sharing of Network Slices and Monetization," in *2023 11th International Conference on Internet of Everything, Microwave Engineering, Communication and Networks (IEMECON)*, IEEE, Feb. 2023, pp. 1–6.

doi: 10.1109/IEMECON56962.2023.10092301.
[153] S. Singh, T. Kumar, A. Pant, and R. Mahalle, "DeepB2B2XSlice Engine - An O-RAN Slice Selection and Resource Optimization AI Engine," in *2022 IEEE 6th International Symposium on Telecommunication Technologies (ISTT)*, IEEE, Nov. 2022, pp. 79–84. doi: 10.1109/ISTT56288.2022.9966553.
[154] S. Arora, A. Ksentini, and C. Bonnet, "Cloud native Lightweight Slice Orchestration (CLiSO) framework," *Comput. Commun.*, vol. 213, pp. 1–12, Jan. 2024, doi: 10.1016/j.comcom.2023.10.010.
[155] PCI-SIG, "Single Root I/O Virtualization and Sharing Specification 1.1 (2010-01)." [Online]. Available: https://pcisig.com/specifications
[156] J. DerGurahian, "Data Center Interconnect (DCI)," 2018. https://www.techtarget.com/searchnetworking/definition/data-center-interconnect (accessed April 15, 2025).
[157] Ciena, "Data Center Interconnect Essentials - Expert Series," 2019. [Online]. Available: https://www.ciena.com/insights/ebooks/data-center-interconnect-essentials.html
[158] 3GPP, "Universal Geographical Area Description (GAD), TS 23.032 V18.3.0 (2024-12)." [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.032/23032-i30.zip
[159] S. Aguzzoli, P. Codara, D. Valota, T. Flaminio, and B. Gerla, "Querying with Łukasiewicz logic," in *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, IEEE, Aug. 2015, pp. 1–8. doi: 10.1109/FUZZ-IEEE.2015.7338061.
[160] S. Kundu and J. Chen, "Fuzzy logic or Lukasiewicz logic: A clarification," *Fuzzy Sets Syst.*, vol. 95, no. 3, pp. 369–379, May 1998, doi: 10.1016/S0165-0114(96)00268-0.
[161] J. X. Salvat, L. Zanzi, A. Garcia-Saavedra, V. Sciancalepore, and X. Costa-Perez, "Overbooking network slices through yield-driven end-to-end orchestration," in *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*, New York, NY, USA: ACM, Dec. 2018, pp. 353–365. doi: 10.1145/3281411.3281435.
[162] E. Pateromichelakis *et al.*, "End-to-End Data Analytics Framework for 5G Architecture," *IEEE Access*, vol. 7, pp. 40295–40312, 2019, doi: 10.1109/ACCESS.2019.2902984.
[163] OASIS, "TOSCA specification Version 2.0, Committee Specification Draft 07," 2024. https://docs.oasis-open.org/tosca/TOSCA/v2.0/csd07/TOSCA-v2.0-csd07.html (accessed April 15, 2025).
[164] R. Parasuraman, T. B. Sheridan, and C. D. Wickens, "A model for types and levels of human interaction with automation," *IEEE Trans. Syst. Man, Cybern. - Part A Syst. Humans*, vol. 30, no. 3, pp. 286–297, May 2000, doi: 10.1109/3468.844354.
[165] European Commission, "5G NORMA (NOvel Radio Multiservice adaptive network Architecture), Horizon 2020 project," 2015. https://cordis.europa.eu/project/id/671584
[166] European Commission, "Novel Enablers for Cloud Slicing (NECOS), Horizon 2020 project," 2017. https://cordis.europa.eu/project/id/777067
[167] Samsung, "Virtualized RAN vol.2 - Technical White Paper," 2021, [Online]. Available: https://images.samsung.com/is/content/samsung/p5/global/business/networks/insights/white-papers/0406_virtualized-ran-vol-2/Virtualized_RAN-Vol.2.pdf
[168] O-RAN Alliance, "O-RAN Architecture Description; O-RAN.WG1.OAD-R003-v12.00 (2024-10)," 2024. [Online]. Available: https://specifications.o-ran.org/download?id=641
[169] A. B. Bondi, "Characteristics of scalability and their impact on performance," in *Proceedings of the 2nd international workshop on Software and performance*, New York, NY, USA: ACM, Sep. 2000, pp. 195–203. doi: 10.1145/350391.350432.
[170] Research and Markets, "5G Network Slicing Market by Infrastructure, SpectrumBand, Segment, Industry Vertical, Application and Services 2025 - 2030," 2025. 5G Network Slicing Market by Infrastructure, SpectrumBand, Segment, Industry Vertical, Application and Services 2025 - 2030 (accessed April 15, 2025).
[171] Grand View Research, "5G Network Slicing market forecast 2023-2030," 2021. https://www.grandviewresearch.com/industry-analysis/5g-network-slicing-market-report (accessed April 15, 2025).
[172] Linux-Foundation, "Nephio: Cloud Native Network Automation," 2025. https://nephio.org/ (accessed April 15, 2025).
[173] RedHat, "What is GitOps?,". https://www.redhat.com/en/topics/devops/what-is-gitops (accessed April 15, 2025).
[174] ITU-T, "ITU-T Recommendation I.130; Method for the characterization of telecommunication services supported by an ISDN and network capabilities of an ISDN," 1988. [Online]. Available: https://www.itu.int/rec/T-REC-I.130-198811-I
[175] 3GPP, "Telecommunication management; Fixed Mobile Convergence (FMC) Model repertoire, TS 32.156, V19.2.0 (2025-03)," 2025. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/32_series/32.156/32156-j20.zip
[176] ETSI, "Network Functions Virtualisation (NFV) Release 3; Information Modeling; UML Modeling Guidelines, GR NFV-IFA 017 V3.4.1 (2020-06)." [Online]. Available: https://www.etsi.org/deliver/etsi_gr/NFV-IFA/001_099/017/03.04.01_60/gr_NFV-IFA017v030401p.pdf
[177] Ericsson, "Network Slicing: Top 10 use cases to target," Stockholm, 2021. [Online]. Available: https://www.ericsson.com/49c6d0/assets/local/networks-slicing/docs/top-network-slicing-use-cases.pdf
[178] Foundation Inc., "What Is a Vertical? (With Examples)," 2025. https://foundationinc.co/learn/verticals/
[179] L. Z. Pedrini, "Radio Physical Resources," 2021. https://www.telecomhall.net/t/radio-physical-resources/11945 (accessed April 15, 2025).
[180] 3GPP, "Management and orchestration; Provisioning; TS 28.531 V18.1.0 (2023-01)." [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/28_series/28.531/28531-i10.zip
[181] R. Rokui, S. Homma, K. Makhijani, L. Contreras, and J. Tantsura, "A Framework for Network Slices in Networks Built from IETF Technologies," Mar. 2024. doi: 10.17487/RFC9543.
[182] B. Wu, D. Dhody, R. Rokui, T. Saad, and J. Mullooly, "A YANG Data Model for the RFC 9543 Network Slice Service," 2025. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-teas-ietf-network-slice-nbi-yang/
[183] X. Geng, L. M. Contreras, R. Rokui, J. Dong, and I. Bykov, "IETF Network Slice Application in 3GPP 5G End-to-End Network Slice," 2024. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-teas-5g-network-slice-application/03/
[184] E. Zeydan, J. Mangues-Bafalluy, J. Baranda, M. Requena, and Y. Turk, "Service Based Virtual RAN Architecture for Next Generation Cellular Systems," *IEEE Access*, vol. 10, pp. 9455–9470, 2022, doi: 10.1109/ACCESS.2022.3144534.
[185] "OpenStack: Open Source Cloud Computing Infrastructure." https://www.openstack.org/
[186] "Kubernetes: Production-Grade Container Orchestration." https://kubernetes.io/
[187] ETSI, "Network Functions Virtualisation (NFV) Release 3;Management and Orchestration; Report on NFV Information Model; GR NFV-IFA 015 V3.4.1 (2020-06)", [Online]. Available: https://www.etsi.org/deliver/etsi_gr/NFV-IFA/001_099/015/

03.04.01_60/gr_NFV-IFA015v030401p.pdf
[188] 3GPP, “Management and orchestration; 5G end to end Key Performance Indicators (KPI); TS 28.554 V19.3.0 (2025-03).” [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/28_series/28.554/28554-j30.zip
[189] M. Fedor, M. L. Schoffstall, J. R. Davin, and D. J. D. Case, “Simple Network Management Protocol (SNMP),” no. 1157. in Request for Comments. RFC Editor, 1990. doi: 10.17487/RFC1157.
[190] R. Enns, M. Björklund, A. Bierman, and J. Schönwälder, “Network Configuration Protocol (NETCONF),” no. 6241. in Request for Comments. RFC Editor, 2011. doi: 10.17487/RFC6241.
[191] A. Bierman, M. Björklund, and K. Watsen, “RESTCONF Protocol,” no. 8040. in Request for Comments. RFC Editor, 2017. doi: 10.17487/RFC8040.
[192] Dell, “Integrated Dell Remote Access Controller (iDRAC),” 2025, [Online]. Available: https://www.dell.com/en-us/lp/dt/open-manage-idrac
[193] Ł. Czekierda, T. Masternak, K. Skałkowski, and P. Wyszkowski, “Applying SOA and virtual organization concepts in an interactive telemedical consultation system,” in *IADIS Multi Conference on Computer Science and Information Systems*, Lisbon, Portugal, 2012. [Online]. Available: http://www.iadisportal.org/digital-library/applying-soa-and-virtual-organization-concepts-in-an-interactive-telemedical-consultation-system
[194] Ł. Czekierda, F. Malawski, and P. Wyszkowski, “Holistic approach to design and implementation of a medical teleconsultation workspace,” *J. Biomed. Inform.*, vol. 57, pp. 225–244, Oct. 2015, doi: 10.1016/j.jbi.2015.08.007.
[195] 3GPP, “NG-RAN; NG Application Protocol (NGAP); TS 38.413 V18.5.0 (2025-03),” 2024. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/38_series/38.413/38413-i50.zip
[196] J. Pieczerak, “First deliverable for NFV Release 5 feature: applying SBA design style to NFV-MANO,” 2022. https://www.etsi.org/newsroom/blogs/entry/first-deliverable-for-nfv-release-5-feature-applying-sba-design-style-to-nfv-mano-1
[197] ETSI, “Network Functions Virtualisation (NFV) Release 5; Architectural Framework; Report on Service Based Architecture (SBA) design, GR NFV-IFA 039 V5.1.1,” 2022. [Online]. Available: https://www.etsi.org/deliver/etsi_gr/NFV-IFA/001_099/039/05.01.01_60/gr_NFV-IFA039v050101p.pdf
[198] L. Suárez, D. Espes, P. Le Parc, and F. Cuppens, “Defining a communication service management function for 5G network slices,” in *European Conference on Networks and Communications (EuCNC)*, Valencia, Spain: IEEE, 2019, pp. 144–148. doi: 10.1109/EuCNC.2019.8802010.
[199] S. Addepalli, “5G & Dynamic Network Slicing,” 2021. https://www.linkedin.com/pulse/5g-dynamic-network-slicing-srinivasa-addepalli (accessed April 15, 2025).
[200] ONAP, “Template Design for Option 1.” https://wiki.onap.org/display/DW/Template+Design+for+Option+1 (accessed April 15, 2025).
[201] ONAP, “Template Design for Option 2.” https://wiki.onap.org/display/DW/Template+Design+for+Option2 (accessed April 15, 2025).
[202] 3GPP, “Telecommunication management; Fixed Mobile Convergence (FMC) Federated Network Information Model (FNIM) Umbrella Information Model (UIM), TS 28.620, V19.2.0 (2025-03).” [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/28_series/28.620/28620-j20.zip
[203] ETSI, “Network Functions Virtualisation (NFV); Management and Orchestration; Report on Architectural Options; GS NFV-IFA 009 V1.1.1 (2016-07).” [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/009/01.01.01_60/gs_NFV-IFA009v010101p.pdf
[204] 3GPP, “Management and orchestration; 5G Network Resource Model (NRM); Stage 2 and stage 3, TS 28.541, V17.7.0 (2022-06),” 2022. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/28_series/28.541/28541-h70.zip
[205] ETSI, “Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification, GS NFV-IFA 005 V4.4.1 (2023-03).” [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/005/04.04.01_60/gs_NFV-IFA005v040401p.pdf
[206] O-RAN Alliance, “O-RAN Work Group 1 (Use Cases and Overall Architecture) Slicing Architecture; O-RAN.WG1.TS.Slicing-Architecture-R004-v14.00 (2025-02),” 2025. [Online]. Available: https://specifications.o-ran.org/download?id=793
[207] ETSI, “Experiential Networked Intelligence (ENI); Study on AI Agents based Next-generation Network Slicing; ETSI GR ENI 051 V4.1.1 (2025-02),” 2025. [Online]. Available: https://www.etsi.org/deliver/etsi_gr/ENI/001_099/051/04.01.01_60/gr_ENI051v040101p.pdf