26-05-2021

White Paper: OAV Architectures

Grant Agreement No.:	856726
Work Package	WP6
Task Item:	Task 2
Document Type:	White Paper
Dissemination Level:	PU (Public)
Lead Partner:	CSUC/RedIRIS
Document ID:	GN4-3-21-292f048

Authors: Daniel Lete (HEAnet), Eduardo Jacob (UPV/EHU-RedIRIS), Hamzeh Khalili (I2CAT/RedIRIS), Iacovos Ioannou (CYNET), Ivana Golub (PSNC), Jasone Astorga (UPV/EHU-RedIRIS), Jerry Sobieski (NORDUnet), Kostas Stamos (GRNET), Maria Isabel Gandia Carriedo (CSUC/RedIRIS), Martin Dunmore (Jisc), Roman Lapacz (PSNC), Simone Spinelli (GÉANT), Sonja Filiposka (UKIM/MARnet), Susanne Naegele-Jackson (FAU/DFN), Tim Chown (Jisc), Yuri Demchenko (UVA/SURF)

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 856726 (GN4-3).

Abstract

The advancement of orchestration, automation and virtualisation (OAV) technologies drives the transition from traditional architectures to agile solutions that can respond to the digital transformation demands. The GN4-3 Network Technologies and Services Development Work Package (WP6), Network Services Evolution and Development Task (T2) has analysed the requirements for building collaborative digital services and proposes a high-level reference architecture that can help the community design and implement their digital platforms in an interoperable way. The flexibility of the proposed blueprint architecture is presented by mapping it against different architectural solutions proposed by standardisation bodies and research projects.

Table of Contents

Executive Summary 1

Introduction 2

Part A Requirements for an Automated Digital Framework 3

A.1 Required Features and Capabilities 4

A.2 Design Principles 5

A.3 Reference Architectures 7

A.3.1 The Open Digital Architecture 8

Part B Mapping Architectural Solutions to the Reference ODA 13

B.1 Mappings 15

B.1.1 Service Provider Architecture 15

B.1.2 MEF Lifecycle Service Orchestration 18

B.1.3 ETSI Zero-touch Network and Service Management 21

B.1.4 ETSI Open Source MANO 24

B.1.5 Open Baton 30

B.1.6 Generalized Virtualization Model (GVM) 32

B.1.7 SENSE 37

B.1.8 Open Network Automation Platform 41

B.1.9 EOSC Architecture 46

B.1.10 ETSI GANA 50

B.2 Architecture Cross-Comparison 54

Part C Mapping Use Cases and NREN Architectures to ODA 59

C.1 Use-Case Mappings 60

C.1.1 5G 60

C.1.2 TALENT 66

C.2 NREN Mappings 70

C.2.1 CYNET 71

C.2.2 SURF 72

Conclusions 73

References 76

Glossary 81

Table of Figures

Figure A.1: ODA functional block grouping based on [ODAFA21] 8
Figure A.2: Orchestration in the Production functional block, based on [ODAIG20] 10
Figure B.1: SPA components mapped to the ODA functional blocks 15
Figure B.2: End-to-end connectivity service [MEF14][MEFLSO] 18
Figure B.3: LSO reference architecture [MEFLSO] 18
Figure B.4: MEF LSO mapped to the ODA functional blocks 19
Figure B.5: ETSI ZSM reference architecture [EZSM] 21
Figure B.6: ETSI ZSM mapped to the ODA functional blocks 22
Figure B.7: ETSI NFV reference architectural framework [ETSINFV002] 24
Figure B.8: ETSI MANO architecture, based on [ETSIMANOA] 25
Figure B.9: Managing WIM functional blocks using Or-Wi reference point, based on [ETSIIFA022] 26
Figure B.10: ETSI OSM mapped to the ODA functional blocks 26
Figure B.11: Example OSM GUI menu 27
Figure B.12: Specifications within the NFV architecture framework [ETSINFVSPEC] 28
Figure B.13: Open Baton architecture release 5 [OB20] 30
Figure B.14: Open Baton mapped to the ODA functional blocks 31
Figure B.15: Virtualisation, management and user control layers in GVM [D8818] 33
Figure B.16: Mapping of GVM and ETSI NFV architectural models 34
Figure B.17: GVM architecture mapped to the ODA functional blocks 35
Figure B.18: SENSE architecture [MON18] 38
Figure B.19: SENSE architectural components mapped to the ODA functional blocks 39
Figure B.20: ONAP architecture [ONAP20] 41
Figure B.21: ONAP mapped to the ODA functional blocks 43
Figure B.22: NBI architecture and its integration with other ONAP components [ONAP20] 43
Figure B.23: EOSC technical architecture – functional view [HTA20] 47
Figure B.24: EOSC technical architecture – interactions between thematic and common services [HTA20] 48
Figure B.25: EOSC architecture mapped to the ODA functional blocks 48
Figure B.26: AFI GANA reference model, taken from [GANA16] 51
Figure B.27: ETSI GANA components mapped to the ODA functional blocks 52
Figure B.28: Main focus of analysed architectures and their common components 54
Figure C.1: 5G overall system architecture [FGPPP16] 61
Figure C.2: 5G architecture mapped to the ODA functional blocks 63

Figure C.3: TALENT architecture 67
Figure C.4: TALENT architecture mapped to the ODA functional blocks 68
Figure C.5: CYNET OAV architecture mapped to the ODA functional blocks 71
Figure C.6: SURF OAV architecture mapped to the ODA functional blocks 72

Table of Tables

Table B.1: Key features, advantages and disadvantages of the analysed architectures 57

Executive Summary

As digital transformations bring new challenges to networking, providers must find ways to automate processes, to ensure agility and technology-agnostic service behaviour. These challenges do not apply to commercial providers only, but also to National Research and Education Networks (NRENs). It is therefore crucial for GÉANT to drive this network evolution using a common strategy for automation, orchestration and virtualisation (OAV) of network services in a multi-domain fashion, bringing all partners on board while allowing for implementation variations and accommodating existing solutions.

This white paper reflects the investigation carried out in the Network Technologies and Services Development Work Package (WP6), Network Services Evolution and Development Task (T2) of the GN4-3 project into a number of advanced network architectures to determine if there were common characteristics that would be a good fit to NREN community networks. The goal was to research whether a blueprint of such common elements could be identified to serve as a recommendation or basic reference structure to streamline OAV efforts and enable future interoperability in the NREN communities as well as with other organisations and systems.

The work looked at a number of architectures from standardisation bodies such as the TM Forum, the MEF Forum and the European Telecommunications Standards Institute (ETSI) standardisation group, and also included findings from recent research such as the Generalized Virtualization Model (GVM), SDN for End-to-end Networked Science at the Exascale (SENSE), Open Networking Automation Platform (ONAP) and European Open Science Cloud (EOSC) architectures. The investigations showed that all these advanced architectural frameworks employ common traits with the aim of facilitating and enabling OAV processing and are crucial for furthering OAV in a multi-domain environment.

These common characteristics include: the decoupling of components and the use of open application programming interfaces (APIs) between them; the separation of user/customer engagement from the core network service management processing; the implementation of a virtualisation layer that allows the definition of service objects independent from the underlying infrastructure or technology domains; and the need for real-time inventories with loop-back information to support resource provisioning. The study identified that the Open Digital Architecture (ODA) by TM Forum not only includes all these common characteristics but is open and flexible enough to serve as a common basis of reference for OAV in GÉANT. The ODA principles are based on decoupling so NRENs can keep their own systems and network services and only share the elements they want to be open through common APIs. As a proof of concept, this study mapped various network architectures to the ODA functional blocks, showing that such a blueprint can indeed serve as a common reference architecture.

This document can be read as a whole, but can also be read in individual parts, starting from the requirements, principles and selected architecture in Part A, then looking at the mapping for one or more architectures of interest in Part B and Part C.

Introduction

The overarching digital transformation is permeating every aspect of digital services, including networking. The demands of the fourth industrial revolution are not confined to the exclusively commercial world of the newly named Digital Service Providers [CSPDT18]; the ways in which the digital transformation affects communications and data processing, and the changes it brings, are also very much in evidence in the research and education sphere. This transformational wave is a strong driver for National Research and Education Networks (NRENs) to investigate how they respond to demand, to rethink their communication with end users, optimise their productivity and operations, and transform the processes needed to provide their services. The envisioned goal includes automated operation centres and smooth scalability, an opportunity for further growth by establishing ecosystems wherein multiple partners can work together in an agile, seamless manner and at the same time provide increased value to their end users.

It is highly desirable that this greater degree of agile collaboration should be based on commonly agreed models and interfaces, which in turn are built upon a common terminology and platform reference architecture blueprint. While cloud-based technologies are seen as key enablers for this transformation, mainly due to their inherent scalability and flexibility, there is another set of technology enablers that need to be considered in the networking environment: network virtualisation for improved efficiency, flexibility and dynamic response; automation, including automated feedback and closed control loops; and orchestration as the next level of seamless integration of separate systems and processes. Of course, fully embracing orchestration, automation and virtualisation (OAV) cannot happen overnight; thus solutions are required in the form of hybrid platform architectures which will successfully manage new, virtual and traditional physical networks.

The aim of this white paper is to provide insight into the existing standards-based blueprints (developed in research projects, open source, and/or accepted by industry), establish a common terminology and approach to information modelling, and outline a proposed path to collaboration in the form of design principles and implementation approaches for OAV architectures for the GÉANT community. However, a common OAV reference framework must be flexible enough to integrate and/or federate other efforts that point towards global R&E interconnectivity, including, but not limited to, the Global Network Architecture (GNA) [GNA20]. It should encourage use of common principles while allowing local, internal variation in how each NREN implements those principles.

This white paper is structured into three parts that can be read separately: Part A contains the main discussion on how OAV efforts could be streamlined using a high-level reference architecture or blueprint, in particular the TM Forum Open Digital Architecture (ODA). This is followed by Part B, which provides an illustration of how a variety of architectures can be mapped to that flexible OAV reference blueprint. Part C investigates commercial and research-based use cases and gives examples of how a large number of various NREN architectures could also make use of this reference architecture, no matter how different their internal processes and workflows may be.

Part A Requirements for an Automated Digital Framework

A.1 Required Features and Capabilities

The orchestration, automation and virtualisation (OAV) architecture focus group within the Network Technologies and Services Development Work Package (WP6) Network Services Evolution and Development Task (T2) of the GN4-3 project has compiled and investigated a list of required features and capabilities that need to be supported by an OAV architecture solution of a research and education (R&E) network ecosystem. This list is by no means exhaustive, or prioritised in any way, and can be used as a supporting cross-reference checklist when deciding on new architectural designs or approaches, technologies or components. The architecture should:

- Implement security by design by addressing all relevant aspects of confidentiality, integrity and availability (CIA).
- Support open application programming interfaces (APIs) and open source implementations whenever possible with strong developer community support and documentation readily available.
- Support interoperability with other industrial platforms, products and standard industry-adopted architectures (see plugtests such as ETSI NFV Plugtest [NFVPLREP] for interoperability checks).
- Support automation in all aspects of service management, including global inter-domain services.
- Support common information and standardised data modelling to represent resources and services.
- Support flexible composite services (based on discrete atomic network service objects) with infrastructure-independent implementation.
- Support service discovery using a service portfolio that provides information about the service classification and inter-service dependencies and other relevant relationships.
- Support organisations and service catalogues of all sizes (large and small).
- Support hierarchical orchestration, including multi-domain orchestration.
- Support implementation of loosely coupled components, i.e. building blocks with a well-defined set of functionalities.
- Support integration of components using popular DevOps or NetDevOps tools and platforms.
- Support abstraction/virtualisation so that the platform can be technology independent and service agnostic, with the possibility to manage resources independently of vendor-specific features.
- Enable flexible scalability (scale in/out and up/down).
- Support innovative future development and research.

In a multi-domain environment with automated provisioning of composite inter-domain services, it is of great importance that the collaborating parties agree to follow a certain minimum set of principles and guidelines. Within the GÉANT community, this multi-domain environment might include GÉANT as an organisation, the NRENs, but also other related parties such as research institutions, universities, etc. In other words, any entity that has its own administrative domain and would like to build an OAV solution to manage its domain and products with their underlying services.

This list of requirements can be used as a checklist of features and capabilities needed to provide an efficient, yet highly agile and dynamic environment that supports a vast set of services built upon physical and virtual resources, as well as enable collaborative efforts with all partners. It can be considered a summary of characteristics or guidelines that should be part of an architecture blueprint or reference architecture for OAV within an organisation, but that at the same time can lay the groundwork for defining a common understanding and common interface between other multi-domain parties. Such an architecture blueprint for OAV is needed to set the foundation for federation, interoperation and collaboration between the separately implemented solutions in the NREN community. Adhering to this minimum set of requirements does not impose any particular restrictions on the manner of implementation of an NREN's architecture; specifics of local implementation are a matter for each NREN. The requirements call for a technology- and service-agnostic design so that any type of technology can be used for the implementation of separate components of the architecture, and the architecture can be used for the agile, model-driven lifecycle management of any type of digital service an NREN would like to offer.

It must be stressed that the requirements and design principles discussed in this section are to be considered as enduring guiding principles that provide a consistent set of rules across the whole organisation and its processes. They are aligned to the standards of The Open Group Architecture Framework (TOGAF) and are based on the TOGAF Architecture Development Methodology [TADM11].

A.2 Design Principles

The design principles that inform the OAV architecture solution are as follows:

Innovation

The OAV goal is to achieve a digital transformation towards an automated architecture, thus the redesign effort should *focus on innovation* together with allowing quick adoption of new technologies and implementing processes that can be fully automated (better and faster), but also changing the ways things are traditionally organised (smarter and more adaptable).

Iteration

However, the transition towards a new OAV architecture should not be seen as a big-bang process. Adopting an approach of *iterative progression with feedback* reduces complexity by following a holistic

roadmap of progressive changes. Lessons learnt from each step help improve the future implementation efforts.

Functional Grouping

Such an iterative progression towards OAV can best be supported by providing a way to *group functional areas* that will guide the *decoupling of components* within the OAV architecture. With a modular OAV architecture where components are based on their functionalities, this grouping provides the freedom to make changes in one functional area which is completely transparent to other areas and facilitates the step-by-step progression towards OAV.

Technology Agnosticism

All architectural *components should be agnostic to the choice of technology* for their implementation. They should be able to be deployed in different technology environments and should have no specific technological requirement. This allows organisations freedom in the specifics of their implementation, while embracing the higher-level principles that will foster better interoperability and collaboration. It allows cost-effective upgrades of components, and transparent changes of underlying technological choices (such as an operating system or type of database).

Component Decoupling

The functional blocks of an architecture can be built using a number of modular components, where *each component is independent* in terms of its origin (build or buy), manner of deployment or location. The component decoupling ensures that all components are autonomous and self-contained when performing their functions.

Exposed Capabilities

The *components should expose their capabilities* via well-defined standard open interfaces described in a components catalogue. In other words, the architecture should enable the automated discovery of components that have a desired capability, thus avoiding any need for manual configuration and increasing independence of current component implementation. All integration of components should be done using *open APIs* whenever possible. Using the same API for both intra- and inter-domain integration further enables an agile approach to the creation of a common ecosystem that rapidly yet uniformly responds to any changes.

Information Governance

The core premise of the architecture includes a common information model that is shared across all components. This requires a well-organised *information governance* that will efficiently manage and protect all relevant information [IG14]. The information governance activities should guarantee the integrity and availability of the information, as well as incorporate actions related to any relevant regulatory requirements. At the same time, data generated in one block should be available to all other relevant components. It needs to be acknowledged that the information can be used for intelligent decision-making supporting different operational decisions on different levels in the organisation. However, sensitive information should be confidential and accessed only using corresponding authentication and authorisation methods. In other words, the architecture should incorporate the “security by design” and “privacy by design” principles on a holistic system-wide level. Having in mind the research and education environment, another important aspect of information

governance is making relevant research data findable, accessible, interoperable and reusable (FAIR) [FAIR18].

Interoperability

The architecture also needs to follow standards that promote *interoperability* for seamless integration across collaborating NRENs, partners, vendors and technologies.

Intent-Based Network Management

One of the future modes of operation that needs to be supported by the new architecture solution is *intent-based network management*. Intent-based networking [IBN18] is aimed at further improving network availability and network agility by being able to automatically translate high-level policies and input from the end users to the necessary network configuration that implements the requirements. In addition, the network behaviour is constantly monitored and if it deviates from the high-level policies, automated actions are taken to correct the identified problems. In other words, intent-based network management implements orchestrated policy management together with automated closed control loops based on feedback from network monitoring and statistics.

A.3 Reference Architectures

In the search for a possible reference architecture that would satisfy the aforementioned requirements, the team has explored a number of architectures from standardisation bodies such as the TM Forum (Frameworx and Open APIs, Open Digital Architecture), the MEF Forum, the ETSI standardisation group (Open Source Management and Orchestration (OSM) and Zero-touch network and Service Management (ZSM), and also included findings from recent research such as the Generalized Virtualization Model (GVM), SDN for End-to-end Networked Science at the Exascale (SENSE), Open Networking Automation Platform (ONAP) and European Open Science Cloud (EOSC) architectures.

It was concluded that the identified requirements and design principles are very much aligned with the philosophy and current efforts of the TM Forum Open Digital Architecture (ODA) project. The main idea of ODA is to offer “an industry-agreed blueprint, language and set of key design principles to follow” [ODAW18]. The complete set of ODA concepts and principles is available in [ODADC21]. Over 50 communication service providers have already committed to the development and implementation of the ODA design principles and best practices. Thus, these design principles and best practices seem very suitable to fit a blueprint that could benefit all NRENs in the GÉANT community with its standardised interfaces and decoupling of components, while at the same time allowing already existing NREN OAV solutions to be integrated into a common network architecture vision. The ODA-defined functional grouping and core modelling rules are used in this white paper to provide a reference point for analysing existing OAV solutions.

A.3.1 The Open Digital Architecture

Following the reference functional grouping as defined in ODA [ODAFA21], the blueprint can be represented using six functional blocks, as given in Figure A.1. This representation enables a high-level view of all enterprise functions that provides an understanding of the focus and responsibilities of each functional block and identification of the points for integration necessary to implement full end-to-end workflows.

Decoupling and Integration

The ***Decoupling and Integration*** block manages the separation of the rest of the functional blocks so that the boundaries between the sets of related functions represented by the functional blocks are respected. This functional block ensures that the integration can be implemented in a flexible way without any predefined combination patterns. The functionalities offered by the Decoupling and Integration fabric include the management of an API catalogue and related documentation, as well as the ability to perform message routing and API mediation, thus enhancing the separated components' APIs with policies, security and control.

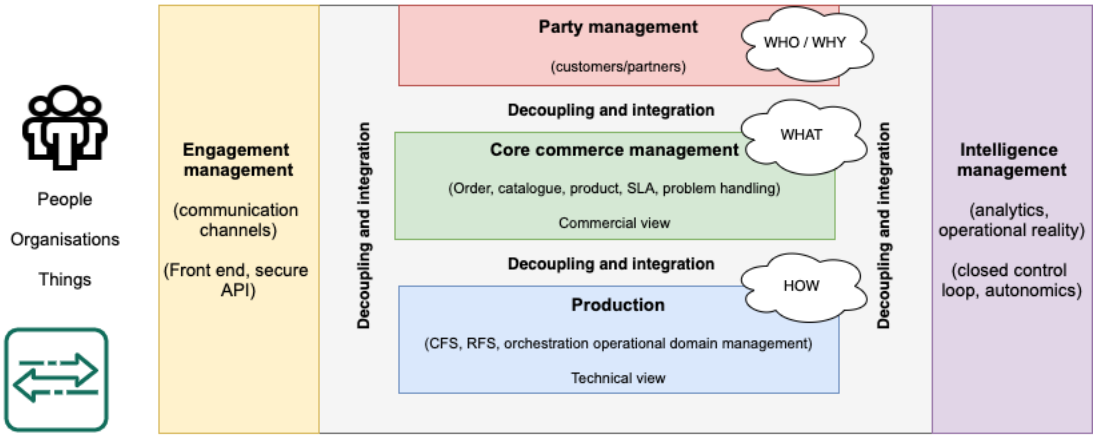

Figure A.1: ODA functional block grouping based on [ODAFA21]

Engagement Management

The ***Engagement Management*** functional block focuses on the interaction with all internal and external actors, which can be people or software agents, customers, employees, partners, third parties, etc. This interaction can be implemented via multiple communication channels. However, the main functionality that needs to be maintained is the omnichannel experience, wherein a user that uses multiple communication channels always experiences a consistent journey. The interactions can include providing information or activating processes and functions that are implemented by components from different functional blocks using the corresponding APIs. It is important to understand that this functional block is a presentation layer only; it does not store any processes, functions or operational data, only technical functions needed to provide the right context to the user. In other words, the Engagement Management functional block is responsible for the front end, authentication and authorisation of users, and management of the user-interaction journeys, including content personalisation and filtering. Another important function of this functional block is

the so-called API HUB that is responsible for exposing a standardised set of APIs to the partners or other external systems.

Party Management

The ***Party Management*** functional block manages the internal and external actors of interest, such as persons or organisations (referred to as Parties). It focuses on management of the information related to the parties, the party roles and rights, and all related marketing, sales and billing activities.

Core Commerce Management

The ***Core Commerce Management*** functional block is responsible for all activities related to the provisioning of main business processes. It manages the catalogue of Product Offers and Products and handles the lifecycle of Product Orders, starting from an order creation and following all of the orchestration actions necessary to fulfil the complete order, including any assurance activities. This functional block is decoupled from the technical concerns of product delivery. This block may also include Product Inventory Management, Problem Handling, Service-Level Agreement (SLA) Management, and other related functions such as Product Testing or Agreement Management.

Intelligence Management

The ***Intelligence Management*** functional block focuses on the processes related to big data analytics using the operational data produced by the other functional blocks. This functional block employs techniques for data analysis, such as trend analysis, correlation and data aggregation needed for network performance evaluation, but also for marketing and sales forecasting. The insight management capability enables this block to uncover new patterns and relations based on historical data, using popular approaches based on artificial intelligence and machine learning. Another functionality of this block is described as Autonomic Manager, which focuses on the implementation of closed control loops that can configure and then adapt in real time the state of a resource or a network using cognitive algorithms. This functional block deals with the knowledge management (including the production knowledge plane) activities on different time scales (fast and slow) to implement self-anything (organisation, healing, tuning, etc.) capabilities. Referring back to the set of design principles, it is evident that the Intelligence Management block includes the implementation of the intent-based networking capabilities and thus should also include the specification of high-level policies and objectives.

Production

The ***Production*** functional block is in charge of service management. Specifically, the products offered in Core Commerce Management are built using the Customer-Facing Services (CFSs), which are exposed by the Production functional block. The CFSs, on the other hand, are built within Production using Resource-Facing Services (RFSs) and Resource Functions (RFs).

Using the exposed APIs of this functional block, the Core Commerce Management block can ask for the fulfilment of services that compose a given ordered product, or the Engagement Management block can forward a user request for changing an existing service. The main functionalities of the Production block include all activities related to the end-to-end service and resource lifecycle management, including – in the case of multi-domain service partnering – actions that may span across multiple organisations that are part of the ecosystem. The Production block is also responsible for the service development, infrastructure deployment, operations management, usage and

performance management, workforce management, resource provisioning, service and resources catalogues and inventories, etc.

The Production functional block enables a flexible solution that can respond with the desired agility, based on the principle of decoupling the process of how services are implemented from the way the production services are offered to the users. This automatically focuses the Core Commerce Management block on a different approach to building products, which is by reviewing what underlying services are offered from the Production block and how they can be combined together to achieve the desired product. At the same time, the differentiation between CFS and RFS enables transparent switching from one technology to another without any changes required above the RFS level or in the exposed services outside the Production block.

Related to the Production functional block is the *Operational Domain Management*, which defines the scope of operations within an administrative boundary (the organisation itself or an external partner) or a technology boundary (services and resources exposed in that domain). The operational domain manages the complete lifecycle of all services and resources that fall into that domain, including service creation, provisioning, termination, assurance, usage and related processes. A *composite operational domain* is defined as an operational domain that creates services based on services offered from other operational domains. In this way, a production service can be defined as a combination of multiple exposed services from multiple operational/technical domains. In addition, the same exposed APIs can be used for a broad range of services (related to connectivity, such as circuits, or endpoint-oriented, such as VMs or applications). This, of course, requires a suitable information model that will enable the high-level abstraction needed to describe all types of services in a common, technology-agnostic way.

These are just some of the reasons why the agility of the new OAV architecture will mostly depend on the choice of implementation of the Production functional block (see Figure A.2). To support the seamless implementation of multi-domain services in a standardised service-agnostic way, common APIs that will expose the Production block capabilities of each domain to the external environment are needed. The internal Production block implementation can be as customised as needed in any organisation.

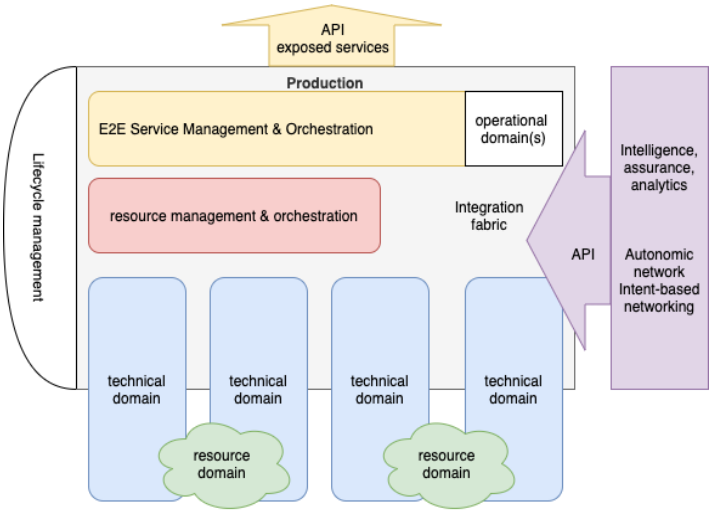

Figure A.2: Orchestration in the Production functional block, based on [ODAIG20]

The Production functional block performs several important functionalities (shown in Figure A.2).

The *End-to-End (E2E) Service Management & Orchestration* functionality is considered to be one of the most crucial in the Production functional block. This is the main provider of the CFSs that are then used by the Core Commerce Management block to build the products, which are in turn defined as a specific combination of one or multiple CFSs. Thus, the CFS (technology agnostic) and RFS (technology specific) management, or, simply, service management, is the core of this functionality, where both types of services can potentially be exposed by the Production block so that products can be built, or multi-domain partnered CFSs can be built.

The implementation of the RFSs is managed by the *Resource Management & Orchestration* functionality, wherein the technology-specific implementation of services is considered. It is important to recognise that within this functionality, the RFSs are still described in a vendor-neutral way. Another type of resource that can be managed in a similar fashion is the Resource Function (RF), which provides an abstraction of resources into functions (heavily used by Network Function Virtualisation (NFV)). The implementation of this component is optional, but recommended in the cases when multi-technology orchestrator tools are used. The functionalities of the Resource Management & Orchestration function component should be exposed via service APIs that deal with RFSs and are to be consumed by the E2E Service Management & Orchestration component.

The vendor- and technology-specific management is captured in the *Technology Management* functionality, which manages the different technology domains that may exist. This is the implementation of the specific management interfaces that depend on the technology and vendor in question. The functionalities that are part of the Technology Management in the technical domains are exposed by APIs that can be consumed by the Resource Management & Orchestration and/or the E2E Service Management & Orchestration component, depending on the implementation. The exact set of functionalities and APIs exposed by the technical domains depends on the nature of the technology and its architectural implementation.

Consumption of the different functional components is done via the *Integration* fabric. There can be more layers of integration depending on the situation and data modelling used. The information modelling of these three layers of abstraction has been a topic of interest for different standardisation organisations, and currently there are a number of models that address all three layers, or a subset, defined by MEF, ETSI, IETF, ONF, the Broadband Forum, NGMN and others.

If intent-based networking is implemented, or any other layer of closed control loops, then there is an integration with the Intelligence Management functional block, which provides the control information in the E2E Service Management where the services are modelled using intent-based modelling patterns. The E2E Service Management can compose CFSs based on existing CFSs, RFSs and RFs exposed from the corresponding function components. By enabling the external information exchange to be only on the CFSs' level, the Production block is empowered to expose resources that otherwise do not expose a resource API. All exposed services are associated with an operational domain, which governs all services and resources within the defined boundary.

The actual deployment of the Production block components that expose the described functionalities depends on many factors which are organisation-specific, and thus the implementation packaging can be done in many different ways. The main goal is that the implementation supports all current network devices, physical and virtual, in a common way. Following the design principles and guidelines, the implementation should support the automated addition of new components over time

in a transparent fashion using a corresponding resource catalogue wherein each component's capabilities are published. Thus, the running environment for the Production functional block can be implemented using a combination of various approaches such as cloud, virtual infrastructure, containers or traditional physical servers.

The separation of CFSs and RFSs is one of the cornerstones on which the Production functional block is built. The customer picks the products from the Core Commerce Management product catalogue, and the CFSs are the services that implement the product. The customer should not have knowledge of the specific RFSs used to support the provisioning of the CFSs. The CFSs are used to describe technology-neutral capabilities and their specification parameters are general, such as latency or SLA parameters. The main function of the RFSs is to provide an abstraction or virtualisation layer on top of the native protocols that are used to manage resources, which allows them to span multiple technical domains. For the virtual resources, there are Resource Functions that model resources as functions. They can also be used to model legacy network appliances, or functionalities of network applications.

The implementation of automation and closed control loops in the Production block can be done on different levels:

- Via E2E Service Management when implementing CFS closed control loops, with the advantage that this approach preserves the E2E service characteristics.
- Via the Resource Management and RFSs, where transparent service changes related to technical events can be made automatically.
- Within the technical domain, with the advantage of very fast reaction to changes and real-time or near real-time reaction.

The policy manager that controls the loops can be implemented as a separate centralised component or distributed in each of the separate functional components where the policy rules propagate down to the lower orchestration layers. The implementation of closed control loops creates a coupling between the Production and Intelligence Management functional blocks, where continuous feedback monitoring and analytics is done in Intelligence Management. Once a decision is made that an action is needed to reconfigure a service or resource, the actual implementation of the necessary changes is made within the Production functional blocks.

The ODA model supports the construction of composite services. The composition is done on the CFS level, where one exposed CFS can be a composite of several other CFSs, which may belong to the same or different operational domains, from one or several organisations. Composition of services originated in different organisations are considered multi-domain services and are thus relevant for the GÉANT community. There are several options for how multi-domain services can be composed through the ODA model. One option is that the E2E Service Management & Orchestration functional component of the requesting operational domain is in charge of the orchestration process of all CFS elements from all related operational domains, with CFSs being mapped into RFSs that abstract the differences in the technical domains. Another option is that RFSs are defined as composite RFSs. Then, it can be realised as composite multi-domain products in the Core Commerce Management block. The final decision would depend on existing systems, service nature and technological considerations of the involved organisations.

This concludes Part A of the white paper. Part B presents a selection of architectural solutions, and maps them to the reference ODA.

Part B Mapping Architectural Solutions to the Reference ODA

An effective way to understand the architectures and implementation of different solutions is by mapping them to an existing, reference architecture. The TM Forum Open Digital Architecture (ODA) has proved to be an excellent option for such a reference architecture, which is why in this section a number of architecture proposals and solutions are presented from the perspective of the ODA functional design, building blocks and functions. The list of investigated architectural solutions has been compiled by the orchestration, automation and virtualisation (OAV) architecture focus group within the Network Technologies and Services Development Work Package (WP6) Network Services Evolution and Development Task (T2) of the GN4-3 project in order to cover diverse approaches, including the most prominent examples given by standardisation bodies, but also some research project proposals that have played a major role in the community. The list of investigated architectures is by no means exhaustive; rather, it presents a sample of the possible paths towards building an OAV architecture. It includes:

- Service Provider Architecture (SPA).
- MEF Lifecycle Service Orchestration.
- ETSI Zero touch network & Service Management (ZSM).
- ETSI Open Source MANO (OSM).
- Open Baton.
- Generalized Virtualization Model (GVM).
- SDN for End-to-end Networked Science at the Exascale (SENSE).
- Open Network Automation Platform (ONAP).
- European Open Science Cloud (EOSC) architecture.
- ETSI Generic Autonomic Network Architecture (GANA).

Note that not all the ODA functional blocks apply to all the architectural solutions investigated.

In addition, the focus group has investigated two use cases that showcase how different components and approaches from a combination of the investigated architectures can be used to design and implement one solution:

- 5G architecture.
- Terrestrial Satellite Resource Coordinator (TALENT).

These, together with two NREN implementations, are presented in Part C Mapping Use Cases and NREN Architectures to ODA.

B.1 Mappings

B.1.1 Service Provider Architecture

The Service Provider Architecture (SPA) [SPAW] is a service management platform that is based on the TM Forum Open Digital Framework. Its development is continued from the previous generation of the GÉANT project (GN4-2) and its Self-Service Portal graphical user interface (SSP GUI) is used for the service order management process of the GÉANT Connection Service (GCS) [SSPM20]. The SPA solution can be used as the basis for the development of a digital platform and to manage the lifecycle of any type of service, including networking, trust and identity or security services.

While all components of the SPA follow the principles of the building blocks architecture, the solution also incorporates other aspects of OAV design principles, such as:

- Ability to compose services and resources.
- Abstract, technology-agnostic modelling of all resources and services.
- Distinction between products, customer-facing and resource-facing services.
- Support for both physical and virtual networks.
- Ability for east-west integration with systems from other domains.

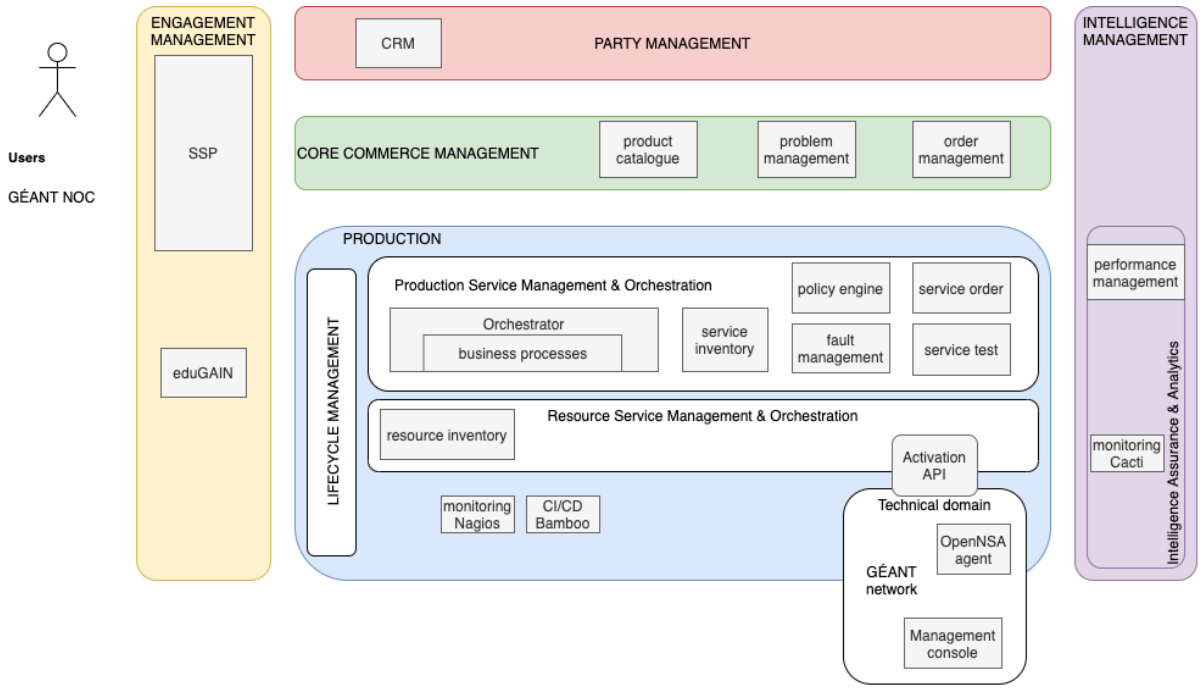

Figure B.1: SPA components mapped to the ODA functional blocks

The SPA design is flexible and easily extended with additional components and features. The component functionalities are orchestrated by a workflow-based orchestrator that can organise

processes in several different groups depending on the workflow level (core management, production service orchestration or production resource orchestration).

Figure B.1 presents the SPA components mapped across the ODA functional blocks. The implementation is also aligned with the ODA concept of a Network as a Service (NaaS) API, which is currently defined as a set of APIs that should be exposed by the Production functional block. The WP6 T2 development team constantly follows the developments of the TM Forum Open APIs and makes corresponding improvements in the SPA implementation.

Engagement Management

The SPA Self-Service Portal, placed in the Engagement Management functional block, is the main GUI for all end users. This GUI is used by the GÉANT Network Operations Centre (NOC) to manage the GCS, and the portal layout and functionalities have been designed in accordance with the needs of this team [SSPM20]. The portal can only read information from other SPA components to provide the requested information to the user. This information is filtered according to the user roles and permissions as they are defined in the customer relationship management (CRM) component in Party Management. All other actions that affect changes in the system are implemented via the Orchestration engine, where user actions in the SSP can trigger processes defined in the orchestration. The SSP does not have any orchestration logic, it is only allowed to call the orchestrator API, while the orchestrator can make changes in the rest of the system.

Access to the portal is only allowed to authenticated users, and for these purposes the portal is integrated with eduGAIN, which provides the authentication and authorisation infrastructure (AAI) functionalities. Once the user goes through the eduGAIN login process, the user credentials are checked with the information in the CRM and, if the user is allowed to use the service, then the corresponding information is presented in the SSP.

Party Management

The main SPA component that belongs to the Party Management functional block is the customer relationship management (CRM). Its implementation is achieved using the SuiteCRM free, open source version, where the native API of SuiteCRM is wrapped with the Customer Management API, defined by TM Forum. The CRM stores all information about all related parties, including end users, organisations and roles. Based on the user role, the complete or limited set of functionalities is provided to the user in the SSP. For example, users with the admin role can perform all available actions, including ordering new services and terminating active services, while users with the view role can only obtain information about the status of current and past services.

Core Commerce Management

The Core Commerce Management functional block contains the SPA components that are related to the product instances managed by users. These include the product catalogue, which also contains information about the way the product is composed of the underlying services, and the product ordering, which captures the initial orders for actions from the user, such as provisioning of a new circuit, changes to an existing circuit or circuit termination. The fault management component enables the capture and handling of user complaints about the systems, or concerning problems with services or other aspects such as SSP problems or party information updates.

Production

The main SPA components fall into the Production functional block, following the ODA orchestration layering principles with service orchestration, resource orchestration and technical domains. Each of the components is exposed via a corresponding TM Forum Open API. The service orchestration layer components include the main service-oriented processes in the orchestrator and all service-related components such as service ordering; service inventory, which stores all service instances; service testing; and service-related fault management. The service and product ordering components are implemented using the free, open source version of Open-Source Ticket Request System (OTRS), where different order types are stored in different queues and references are created between related tickets. Similar implementation is used for the ticketing related to problem management and fault management, where, in the case of fault management, the ticket creation is based on an alarm raised by a service monitoring system. The service test component is used to perform regular tests of the service. In addition, the service test component can be used to perform a pre-release check before informing the user that an order is feasible. This regular testing may be integrated with an external system for performance management. There is also an implementation of a policy engine, which acts as a service qualification component, where, based on the defined rules, the engine checks if a service with the requested parameters can be ordered or otherwise managed by the requesting user.

Although in Figure B.1 the orchestrator is mapped only to the production service orchestration layer, some of the processes belong to the Core Commerce Management functional block, while others are part of the resource orchestration layer. The resource orchestration layer also includes an in-house implementation of a resource inventory, which contains the logical abstract representation of all resources that are managed by the system. The referencing between the service and resource inventory enables continuous tracking of the resource usage and all service/resource dependencies. This inventory must be treated as a single source of truth so that the information is consistent across the system and the orchestration processes can be implemented in a fully automated fashion. At the moment, all processes defined in the orchestrator are completely automated; no human intervention is needed other than the initial request trigger. However, if necessary, the orchestrator can also support partial automation, involving multiple human actions during the process.

The Production integration fabric includes also: a component monitoring system based on Nagios, a continuous integration / continuous development (CI/CD) pipeline for a DevOps approach to the development of each separate component and process in the orchestration engine based on Bamboo, and other supporting elements such as BitBucket.

The Production domain interfaces with the GCS technical domain via the OpenNSA agent and its management console station. This is achieved via the TM Forum Activation and Configuration Open API, which can be used to provision new circuits, query or change the status of a provisioned circuit, and terminate a circuit, and the results are stored in the service and resource inventory as appropriate.

Intelligence Management

Monitoring in the SPA is activated as a part of a new circuit provisioning process and is then deactivated within the circuit termination process. At the moment, no automated control loop or alarm is implemented.

B.1.2 MEF Lifecycle Service Orchestration

Metro Ethernet Forum Lifecycle Service Orchestration (MEF LSO) [MSOS16] is a layered abstraction architecture defined to build the systems for coordinated management and control across all network domains responsible for delivering an end-to-end connectivity service. LSO provides interoperable automation of operations over the entire lifecycle of Layer 2 and Layer 3 connectivity services, as shown in Figure B.2. This includes design, fulfilment, control, testing, problem management, quality management, billing & usage, security, analytics and policy capabilities. The architecture specifies high-level functional requirements, the functional management entities and the management interface reference points (logical points of interaction) between them.

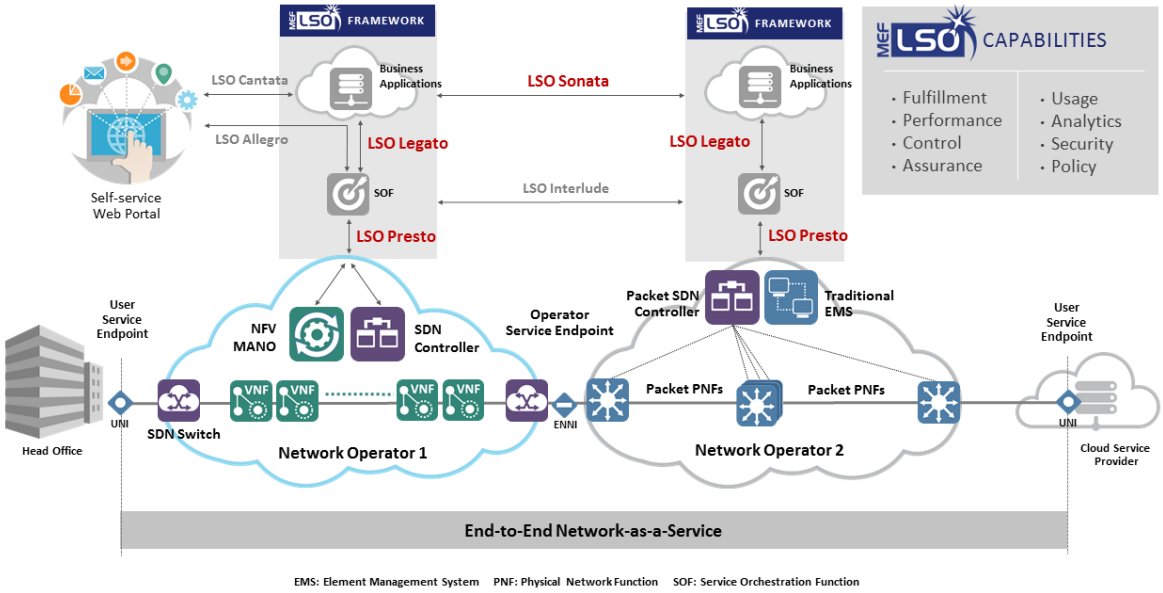

Figure B.2: End-to-end connectivity service [MEF14][MEFLSO]

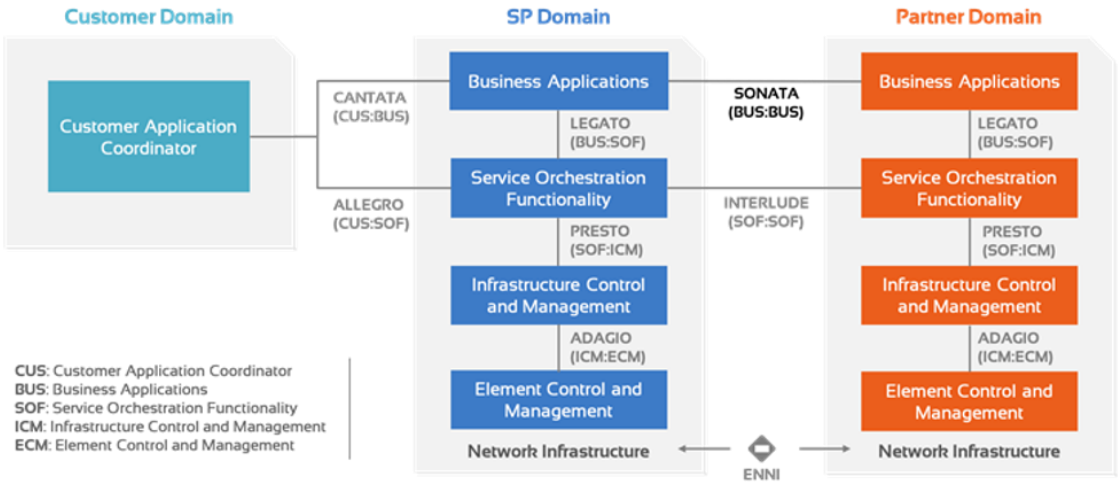

Figure B.3: LSO reference architecture [MEFLSO]

High-level operational threads, shown in Figure B.3, describe orchestrated connectivity service behaviour and interactions among the entities.

The LSO architecture has been proposed to design the management solutions for the Third Network, which addresses new network requirements and trends being observed in the current evolution and transformation of network services. The Third Network combines the on-demand agility and ubiquity of the Internet with performance and security assurances. Distinction between physical and virtual resources becomes transparent. The Third Network vision is based on Network as a Service (NaaS) principles, which make the network appear as a user's own virtual network and enables the user, dynamically and on demand, to create, modify and delete services via customer web portals or software applications.

Mapping of the MEF LSO architecture to the reference ODA is presented in Figure B.4.

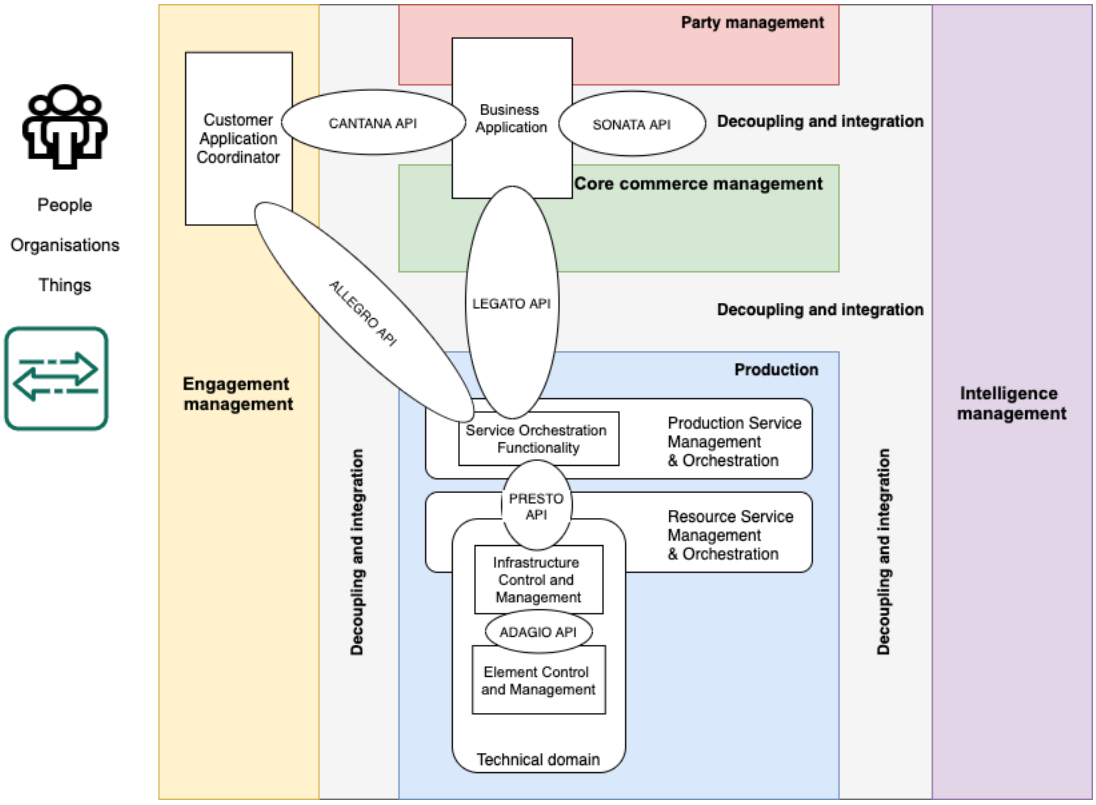

Figure B.4: MEF LSO mapped to the ODA functional blocks

Engagement Management

MEF LSO defines a functional management entity called Customer Application Coordinator (CUS) in the Customer domain which is responsible for coordinating the management of the various service needs (e.g., compute, storage, network, etc.) of specific applications. The CUS supports Customer interactions with the Service Provider to request, modify, manage, control and terminate the use of Products.

The CUS makes use of two interface reference points. The first one, CANTANA, connects to the Service Provider's Business Applications (BUS) entity with capabilities to support the operations interactions

such as ordering, billing, trouble management, etc. The second point, ALLEGRO, allows the CUS supervision and control of dynamic service behaviour through interactions with the Service Provider's Service Orchestration Functionality (SOF). ALLEGRO may also be used to share service level fault information with the Customer.

Party Management

The Business Applications functional block spans two domains: the Party Management domain for billing and relationship management and the Core Commerce Management domain with product offerings. It interfaces CUS in the Engagement Management domain using the CANTANA API, and uses the LEGATO interface reference point to forward instructions based on a customer order to the Service Orchestration Functionality, for example, to instantiate a connectivity service. LEGATO may ask the BUS to send an order to a partner Service Provider to get a service needed as a service component of an end-to-end connectivity service. For compound services, SONATA is used to connect two service providers which cooperate to offer, for example, a multi-domain connectivity service.

Core Commerce Management

The already mentioned BUS entity covers core commerce management functionalities such as product catalogue and ordering. Also, the MEF's specification of product inventory management (MEF 81) [PIMM19] may be used for the lifecycle product management. The specification supports the requirements defined in the MEF architecture requirements for inventory over the SONATA interface (interactions between Service Provider and Partner). At present, the document is limited to the business process requirements depicted as use cases and attribute definitions. It will be the basis of requirements for a Product Inventory Information Model, Data Model and API.

Production

Three functional management entities, Service Orchestration Functionality (SOF), Infrastructure Control and Management (ICM) and Element Control and Management (ECM), fit into the Production block. The SOF provides functions for automation of the service lifecycle (design, fulfilment, control, testing, problem management, quality management, usage measurements, security management, analytics, and policy-based management). The ICM contains the set of functionalities providing domain-specific network and topology-view resource management capabilities, including configuration, control and supervision of the network infrastructure. The ECM contains a set of functionalities supporting element management layer capabilities for individual network elements.

For the communication between SOF, ICM and ECM, the MEF has defined additional management interface reference points: PRESTO and ADAGIO. If Partner services are used, independent SOF components may exchange information. Through the INTERLUDE point, the SOF may request initiation of technical operations or dynamic control behaviour associated with a service in a Partner network domain.

Intelligence Management

The MEF LSO [MEF55] formulated a set of functional requirements for analytics capabilities in LSO management entities. LSO shall support the fusion and analysis of information across domains; ensure information is visible and accessible when and where needed; assemble complete operational pictures of services, service components, physical and virtual infrastructure; and support trending and prediction of service and resource demand and development.

B.1.3 ETSI Zero-touch Network and Service Management

The European Telecommunications Standards Institute (ETSI) Zero-touch network and Service Management (ZSM) working group has the aim of defining a universal framework to enable end-to-end zero-touch service automation involving different management domains. The proposed framework [EZSM] focuses on closed-loop service automation based on monitoring and high-quantity data collection, to which machine learning and artificial intelligence procedures apply. The end-to-end automation procedures considered by ETSI ZSM comprise all operational processes and tasks, including delivery, deployment, configuration, lifecycle management, assurance, optimisation and finalisation. It aims to provide 5G E2E network slicing and management capacity, ideally with 100% automation or zero human intervention, and aims to be general enough to integrate current and future networks and services.

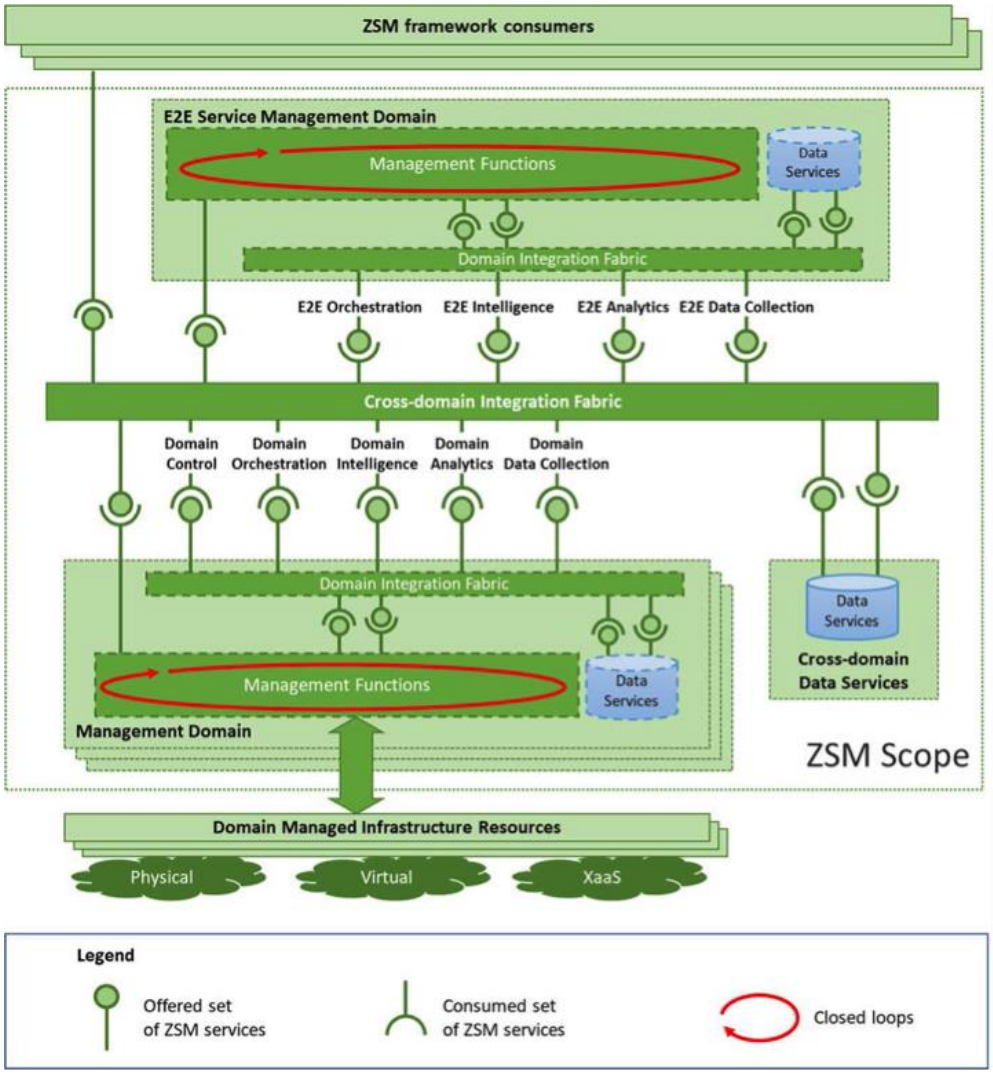

Figure B.5: ETSI ZSM reference architecture [EZSM]

ETSI is promoting different standardisation initiatives (Network Function Virtualisation (NFV), Multi-access Edge Computing (MEC), Experiential Networked Intelligence (ENI) and Open Source MANO

(OSM)) and follows an open approach to allow agile, efficient and qualitative service virtualisation and orchestration within a single management domain. Each of these standardisation initiatives focuses on dedicated aspects of network and service management. The reference architecture defined by the ETSI ZSM working group is graphically depicted in Figure B.5.

Mapping of the ETSI ZSM reference architecture to the ODA architecture has already been performed by a joint effort of TM Forum and ETSI [EZSML]. According to this mapping, most of the ETSI ZSM framework elements correspond with the Production layer in ODA, as shown in Figure B.6. Even though it could be argued differently, for example, that the ETSI ZSM architecture contains elements that fit also into other ODA domains, this white paper keeps the mapping agreed between TM Forum and ETSI.

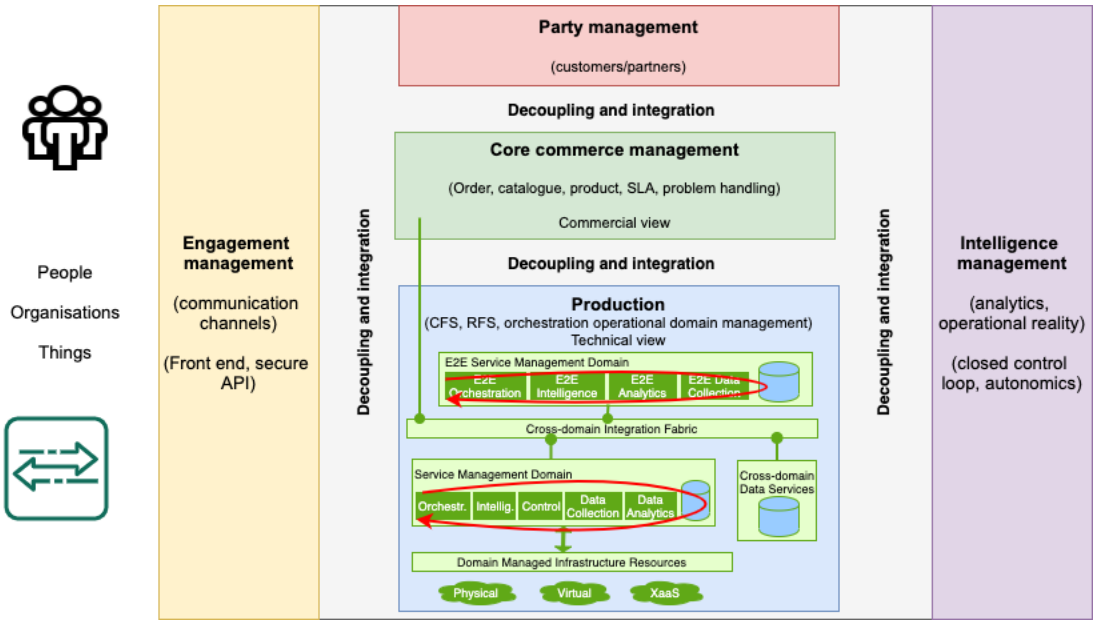

Figure B.6: ETSI ZSM mapped to the ODA functional blocks

Core Commerce Management

External customers interact with the ETSI ZSM framework through the Cross-domain Integration Fabric layer, which supports intent-based interfaces. The management services provided by the ETSI ZSM framework to external customers might be consumed by either human or non-human third parties, such as digital storefronts, web portals, business support system (BSS) components or other ZSM framework instances. Therefore, ETSI ZSM provides both machine-consumable interfaces and the possibility to implement GUIs, web portals or other applications to facilitate access to ZSM services to human customers. Apart from external customers, ZSM services provided by individual operational domains can also be consumed by some other individual operational domains or by the E2E service management domain.

Once the E2E service is deployed, the E2E performance data reporting service might be used to monitor the performance of the E2E service and check if SLAs are met. As in the case of the previous E2E services, in order to obtain performance data regarding the E2E service, the E2E performance data reporting service makes use of the performance data reporting services of the involved individual operational domains, gathers the data obtained from each individual operational domain and creates

an aggregated report. Other services that could be used to monitor the performance of the E2E service are the E2E service condition detection service, which allows KPI thresholds to be set at the E2E service level; and the E2E testing service, which allows testing of whether the E2E service has been successfully deployed. As before, the E2E services rely for their operation on the corresponding operational domain-level services provided by the involved individual operational domains.

Production

In the context of the ETSI ZSM framework, the offered E2E services might consist of different individual operational domains. To achieve the deployment and management of these E2E services, the central element of the ETSI ZSM framework is the Cross-domain Integration Fabric, which provides procedures to interact with the individual operational domains on one hand, and with the end-to-end service management domain on the other. That is, this layer acts as an intermediary that allows end-to-end service and network management services to communicate with individual operational domains in order to make use of the specific procedures provided by each operational domain and, in this way, be able to provide a cross-domain end-to-end solution. The services towards the E2E service management domain that maps to the Production domain of the ODA include the following:

- E2E service orchestration: gathers all the services relating to the coordination of the provisioning, configuration and lifecycle management of cross-domain E2E services provided to external consumers.
- E2E service intelligence: includes services focused on enabling automated decision-making in the E2E cross-domain services by means of closed-loop automation.
- E2E service analytics: includes services that deal with E2E service-related KPIs.
- E2E service data collection: includes services relating to the collection of data (currently performance data) regarding the cross-domain E2E service.

For their operation, most of these services rely on corresponding services provided at the individual operational domain level, namely, domain control, domain orchestration, domain intelligence, domain analytics and domain data collection.

In order to gather available E2E services, the ZSM E2E service orchestration service includes an E2E services inventory management service, which collects information about available E2E services from individual operational domains, and an E2E services inventory information service, which provides information about available E2E services through a northbound interface. For their operation at the E2E service management domain, these services make use of the corresponding services at the individual operational domain level.

Additionally, the ZSM framework also provides its customers with a set of management capabilities for managing E2E services. The catalogue of supported E2E management services is maintained by the managed services catalogue management service within the E2E service orchestration service, in the E2E service management domain. The managed services catalogue management service maintains a catalogue of service models and exposes it to the ZSM framework consumers. Service models include service templates, coverage areas, associated SLAs, stage of the lifecycle, etc. Additionally, service models might be grouped in service categories.

At the individual operational domain level, each domain maintains an inventory of network services and virtualised resources managed by the given domain, as well as an updated view of the operational

domain topology. This is managed by means of the domain orchestration service, which makes use of two further services:

- Domain inventory information service: implements query/response-based mechanisms to provide information about available infrastructure resources and managed services. Information might be provided to customers within the management domain and to the E2E management domain.
- Domain inventory management service: collects information about available infrastructure resources and managed services. This service is only available to customers inside the given operational domain.

Finally, data services implement different storage mechanisms to guarantee data persistence and to enable efficient data sharing among authorised consumers, both at the individual operational domain level and at the cross-domain E2E service management domain level. In this context, data privacy and security aspects receive special consideration within the ZSM framework.

B.1.4 ETSI Open Source MANO

ETSI Open Source MANO (OSM) [EOSM20] is a management and orchestration solution that complies with the ETSI NFV architectural framework. While it offers its own Virtual Infrastructure Manager (VIM), it also supports many third-party VIMs such as OpenStack, VMware vCloud and VIO, Amazon Web Services, Azure and Google Cloud and, since release seven, also Kubernetes. This last integration allows the deployment of more than 20,000 pre-existing production-ready Kubernetes applications, with no need of any translation or repackaging, which are added to the steadily increasing number of Virtual Network Functions (VNFs) commercially or freely available.

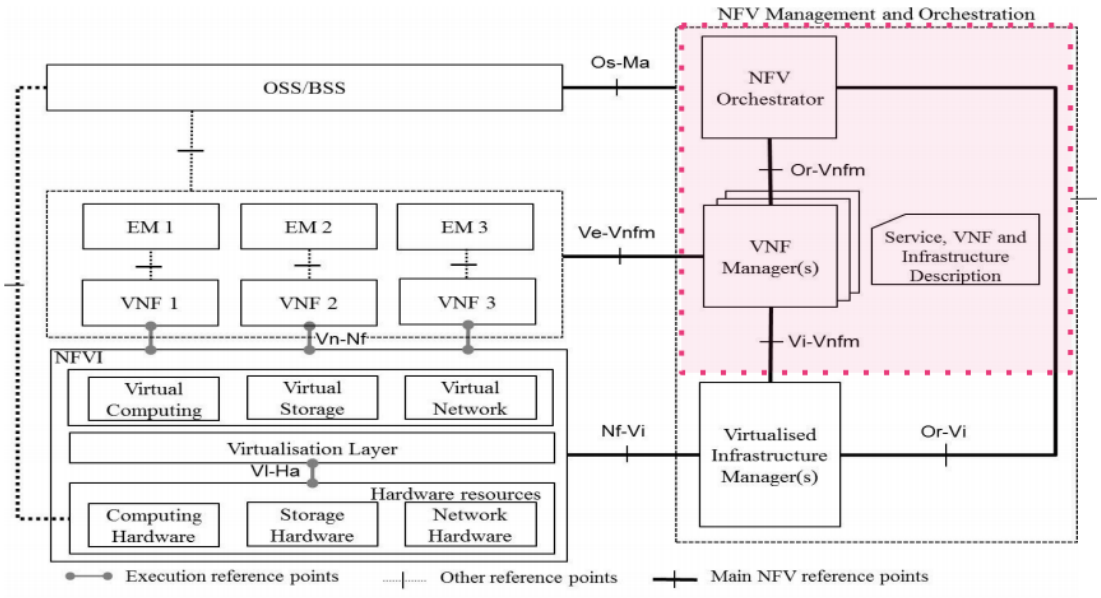

Figure B.7: ETSI NFV reference architectural framework [ETSINFV002]

ETSI OSM is composed of 689,914 lines of code produced by 141 contributors under the rather permissive Apache 2.0 licence. This has allowed the appearance of several commercial distributions baked and supported by companies such as Canonical, Whitestack and Tata.

ETSI OSM covers the two upper layers of the ETSI NFV architectural framework's NFV Management and Orchestration, as shown in Figure B.7. It is important to note that although the VIM is normally part of the MANO, in many cases it is a part of the Network Function Virtualisation Infrastructure (NFVI).

The other two blocks fulfil the following tasks:

- The NFV Orchestrator (NFVO) is in charge of the orchestration and management of NFV infrastructure and software resources, and of realising network services on NFVI. For this it supports Virtual Network Functions (VNFs) and Network Services (NSs) catalogues, and NFVI resources if used (which eliminate the need to talk directly to the northbound interface of the VIM).
- The VNF Manager (VNFM) is responsible for VNF lifecycle management (e.g. instantiation, update, query, scaling, termination). Multiple VNF Managers may be deployed; a VNF Manager may be deployed for each VNF, or a VNF Manager may serve multiple VNFs. Actual mechanisms that can be used to implement VNF Managers are Juju, Python, Ansible and Helm. OSM supports Day-0, Day-1 and Day-2 configuration.

In turn, OSM is composed of the software architecture shown in Figure B.8.

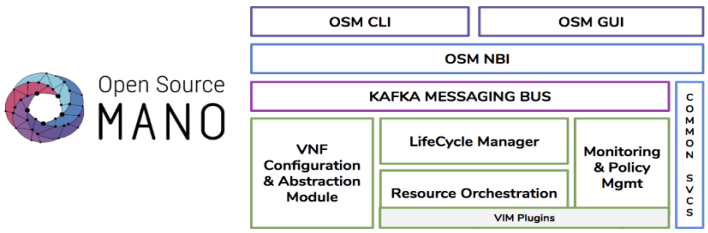

Figure B.8: ETSI MANO architecture, based on [ETSIMANOA]

The main components are:

- LifeCycle Manager (LCM) – the general orchestrator of Network Services.
- Resource Orchestration (RO) – the component in charge of the lifecycle of resources at the VIM layer, so it interfaces with virtualisation managers (such as OpenStack) and SDN controllers.
- VNF Configuration and Abstraction Module (VCA) – the component in charge of communicating with the VNFs, mainly for Day-2 configurations, achieved today using a Juju controller in native or proxy mode (which allows the use of mechanisms such as SSH, RPC or REST through a Python code stub).
- Monitoring Module (MON) – the component that collects metrics from different VIMs, and also VNFs.
- Policy Module (POL) – the component that, given a particular VNF status or metric value, configures alarms and implements actions, such as auto-scaling.

The NFVO can also use a WAN Infrastructure Manager (WIM), which allows the interconnection of the separated WIM in a dynamic and configurable way. The WIM's API aims to be largely independent of the specific underlying elements, the network topology underneath and/or the switching technology itself.

Through SDN/SDTN controllers it is possible to set up many different types of connection technologies: virtual networks for a VIM, MPLS connections, VPN connections (overlay or with interaction with physical equipment), inter-DC connections (various types), MAN connections, etc. OSM follows a plugin model for the inter-VIM connection calls, which supports ONF Open Transport API (TAPI), ONOS and DynPaC. This makes use of the Or-Wi interface, which is a subset of the Or-Vi interface, shown in Figure B.9.

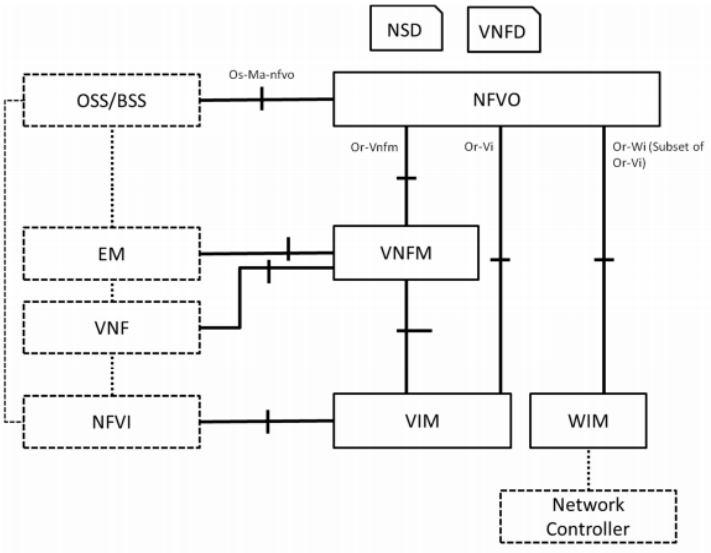

Figure B.9: Managing WIM functional blocks using Or-Wi reference point, based on [ETSIIFA022]

The mapping of OSM to ODA is presented in Figure B.10.

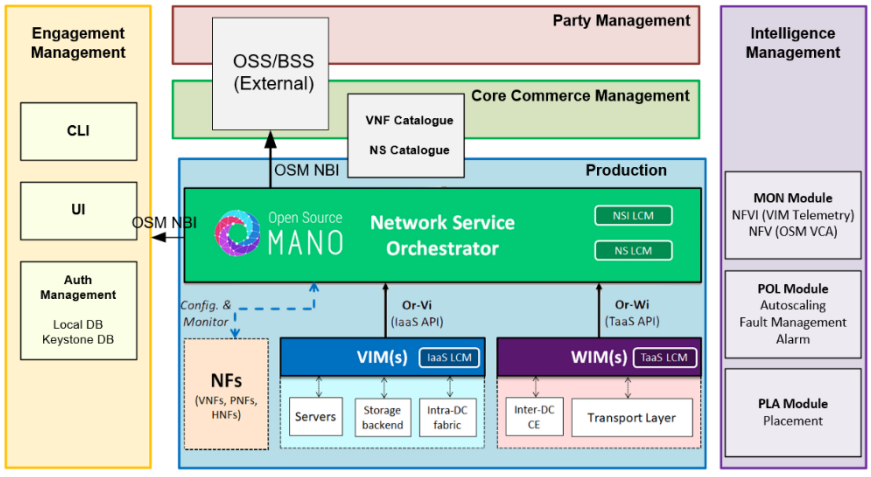

Figure B.10: ETSI OSM mapped to the ODA functional blocks

Engagement Management

The OSM architecture provides a general GUI (Figure B.11) with several features that enable the onboarding of Network Service (NS) packages, a Network Slice Template and Virtual Network Function (VNF) packages, as well as the high-level management of users, projects and roles, and other kinds of resources. The catalogues of Network Slices, NSs and VNFs are accessible through the GUI. The descriptor included in a package can be further edited through the GUI, either in text format (YAML or JSON), or in a graphical editor. Additional artifacts of the package, such as cloud init scripts or charms in a VNF, are provided during the onboarding of the package.

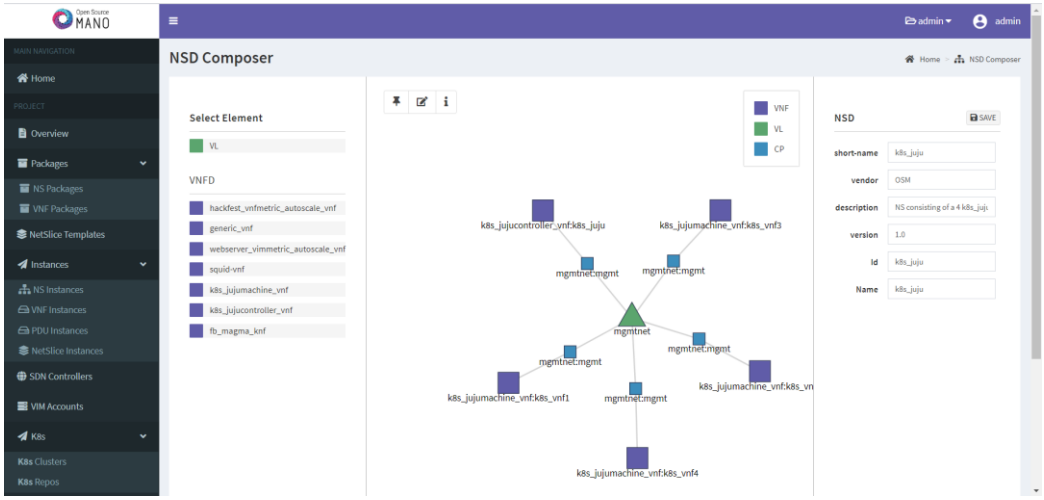

Figure B.11: Example OSM GUI menu

The repository of VIM accounts, SDN controllers, WIM accounts and Kubernetes clusters can also be created and managed through the GUI. The required information and credentials of such resources can be entered into the corresponding form during their creation.

The OSM command line interface (CLI) enables the execution of lower-level operations to complement what can be done through the GUI. Primarily, it introduces more convenient features related to the lifecycle management of VNFs and NSs, such as the execution of scaling operations and the creation of alarms.

OSM also provides a REST API, known as the Northbound Interface (NBI), which is aligned with the ETSI NFV SOL005 standard. This allows an external system in the OSS/BSS layer of the ETSI NFV architecture to communicate and perform operations towards OSM, as specified by the SOL005 standard. In this way, operations regarding the creation, onboarding and lifecycle management of resources, as well as information retrieval, can be carried out with HTTP requests.

One of the strengths of ETSI OSM is that it closely follows many of the interfaces described in ETSI NFV standards (see Figure B.12). Moreover, this is regularly tested in ETSI NFV Plugtests. The latest report, which covers the 4th NFV Plugtests activities, assesses the level of conformance of participating Functions Under Test of VNFs, VNFMs and NFVOs (operated by participants) with NFV SOL002, SOL003 and SOL005 APIs and Open APIs. Additionally, in the latest Plugtests, a first experimental “MEC Interoperability Track” was held. The MEC Track consisted of three types of interoperability tests

covering routing, application lifecycle and APIs for Multi-access Edge Computing solutions. The report is available at [NFVPLREP].

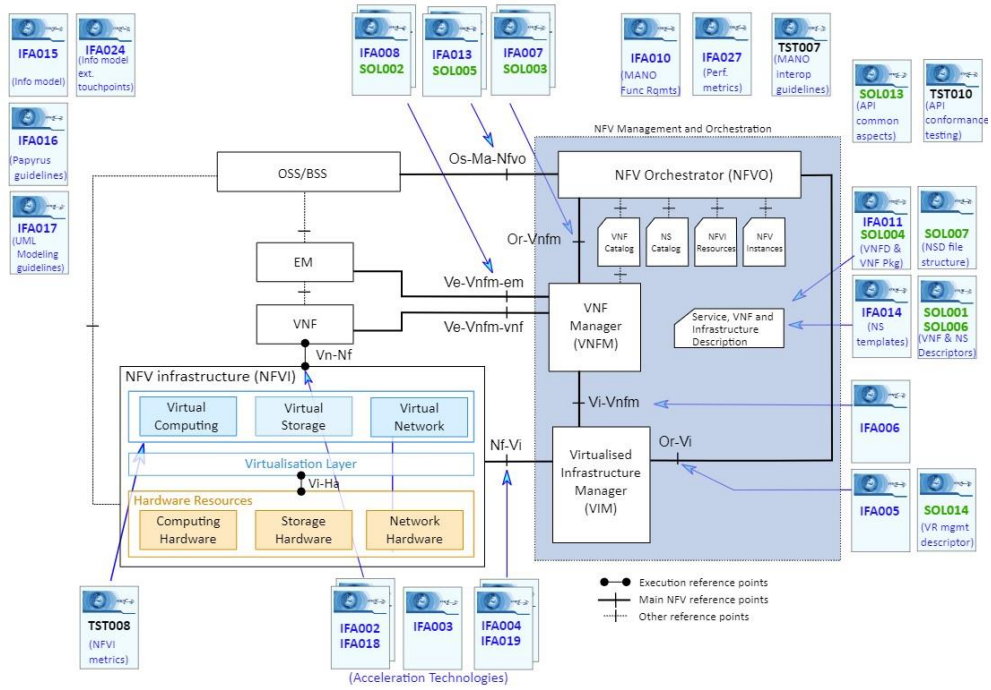

Figure B.12: Specifications within the NFV architecture framework [ETSINFVSPEC]

The ETSI OSM information model (available as a YANG model in the OSM repository) includes the following models, of which OSM is capable of automating the full lifecycle:

- VNF Descriptor (VNFD) and VNF Record (VNFR) which represents the state of a VNF running instance.
- Network Service Descriptor (NSD) and the corresponding Network Service Record (NSR).
- Network Slice Template (NST) and the corresponding Network Slice Instance (NSI).

The NBI interface is based on ETSI NFV SOL005; its description is available in Swagger [ENSS20]. This interface makes it possible to integrate ETSI OSM into existing OSS/BSS systems or even into a Metro Ethernet Forum architecture.

Core Commerce Management

OSM, as its definition implies, does not have OSS/BSS functionalities, but there are at least three commercial solutions from NEC (through its subsidiary Netcracker), MYCOM OSI from the Assurance Cloud Company, Tata Elxsi's TE-OSM and one open source OSS/BSS, Openslice [OSLICE].

The OSM VNFs, NSs and slices catalogues are supplemented by a high-level services catalogue. There are third party applications that can use Openslice through TM Forum Open APIs. As a result, orchestration of customer- and resource-facing services (CFSs/RFSs) that are not natively part of ETSI OSM can be managed at this level.

Party Management

The users can be either OSM users making direct use of the OSM interfaces or users of a third-party OSS/BSS such as Openslice. OSM user management includes the concept of projects, users (assigned to one or more projects) and roles. A user can create a service package (or obtains it from a third party) and uploads it to the OSM catalogue. The package then becomes accessible within the domain of the project where it was uploaded, including for all the users that have access to that project.

Production

An OSM service includes at least a Network Service package, describing the E2E service, and one or more Virtual Network Function packages, describing the components of the service. The service can then be instantiated at any time, on demand. When it is instantiated, the resulting Virtual Deployment Units (VDUs) and/or Container Deployment Units (KDUs) are created.

The orchestration of both physical and virtual components is supported in OSM. The types of technology supported are Virtual Network Functions (VNFs), Physical Network Functions (PNFs), Container Network Functions (CNFs) and Hybrid Network Functions (HNFs). HNFs may contain any number of VDUs, Physical Deployment Units (PDUs) and/or KDUs. PDUs correspond to existing physical equipment, which often provides a networking function such as a router, a firewall or a load balancer.

The supported VIM technologies include OpenVIM, OpenStack, VMware, AWS and OpenNebula. The implementation of the virtualisation layer and hypervisor is managed by the VIM and OSM is technology agnostic in this regard. The deployment of resources in the infrastructure is ultimately managed by the VIM, but some configuration can be provided in the VNF Descriptors to infer the placement of resources. By using different VIM accounts, it is also possible to have different domains within the VIM, depending on the OSM user and/or project. With regard to SDN controllers, the supported technologies are ONOS, OpenDaylight and Floodlight. With regard to WIMs, ONOS, TAPI, OpenDaylight and DynPaC are supported.

Regarding the access to the resources, OSM does not need to have exclusive access to resources. OSM is designed to “politely” request resources from the platforms below. By default, it assumes that there may be other users/tenants requesting resources from each VIM/WIM, etc. Nevertheless, if the application needs to be the only one to which resources from a given VIM target can be assigned, it can also support the model.

Intelligence Management

OSM allows monitoring of the VNFs, which includes the collection of metrics and the automation of scaling operations based on the metrics gathered. Furthermore, OSM also includes support for the monitoring of the health and status of the OSM components. Separate modules are in charge of Monitoring (MON) and Policy (POL) tasks. The monitoring stack includes a Prometheus time series database (TSDB) and exporter for metrics collection, and a Grafana service for visualisation. Metrics can either be collected from the VIM at VDU level, or using Juju at VNF level. Meanwhile, policy functions are responsible for the evaluation of monitored metrics in order to trigger execution of closed-loop operations, which include auto-scaling and alerting. Policy functions are also available at VDU granularity.

On the other hand, the Placement component of OSM is responsible for the optimisation of the placement of VNFs when multiple VIMs are available. The Placement module can automatically place the different VNFs in the optimal VIMs, taking into account parameters and constraints such as the cost of computing in each VIM, the cost of each networking link between VIMs, VNF-pinning constraints, and networking constraints such as latency and jitter. The Placement engine works by selecting the cost-optimal distribution of VNFs that fulfil the specified constraints.

B.1.5 Open Baton

Open Baton [OB20] is an open source platform developed by Fraunhofer FOKUS and the Technical University of Berlin that enables implementation of an NFV environment based on the ETSI NFV MANO architecture framework [ETSINFVSPEC]. The latest release is version 5 from November 2019, available on GitHub [OBGH], which provides a large number of components, shown in Figure B.13.

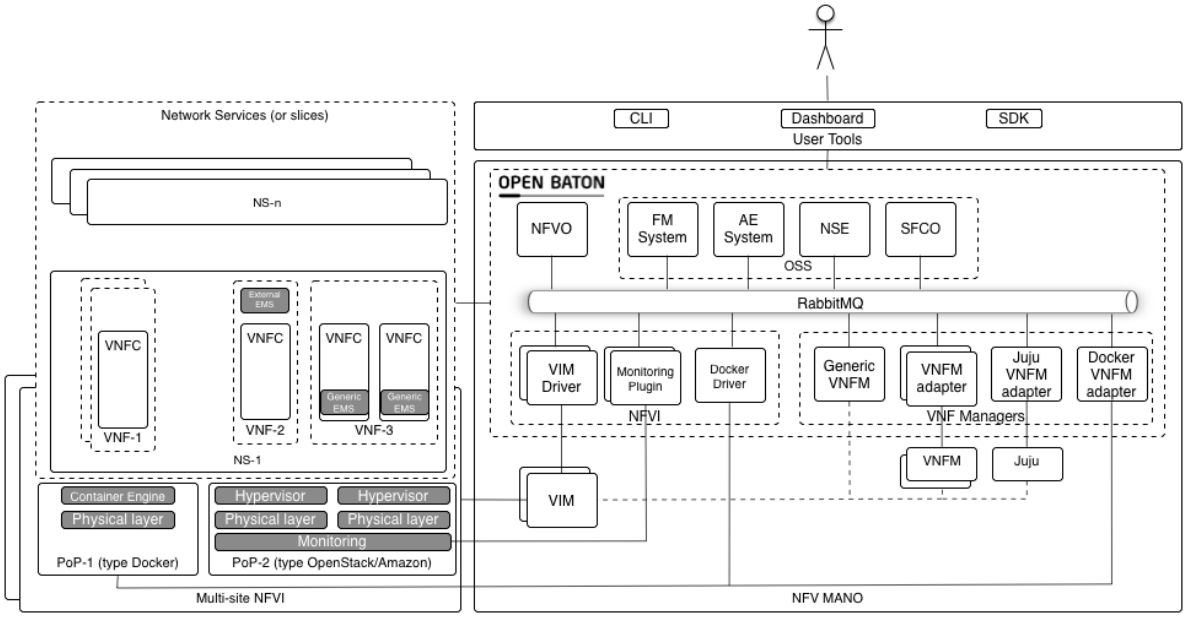

Figure B.13: Open Baton architecture release 5 [OB20]

Open Baton uses an agile development process to provide a customisable network service orchestration framework based on the MANO NFVO. The framework supports multiple sites that provide heterogeneous Network Function Virtualisation infrastructure. Using the generic Virtual Network Function Manager (VNFM) the architecture can manage a large set of different VNFs. In addition to the already developed adapters, the architecture can be further extended with newly developed adapters. It supports multi-tenancy using the network slicing feature and it implements all aspects of operational management such as monitoring, fault management, etc. The tools provided for user access and management include a command line interface (CLI), software development kit (SDK) and a dashboard.

The mapping of the Open Baton implemented components to the ODA functional blocks is presented in Figure B.14.

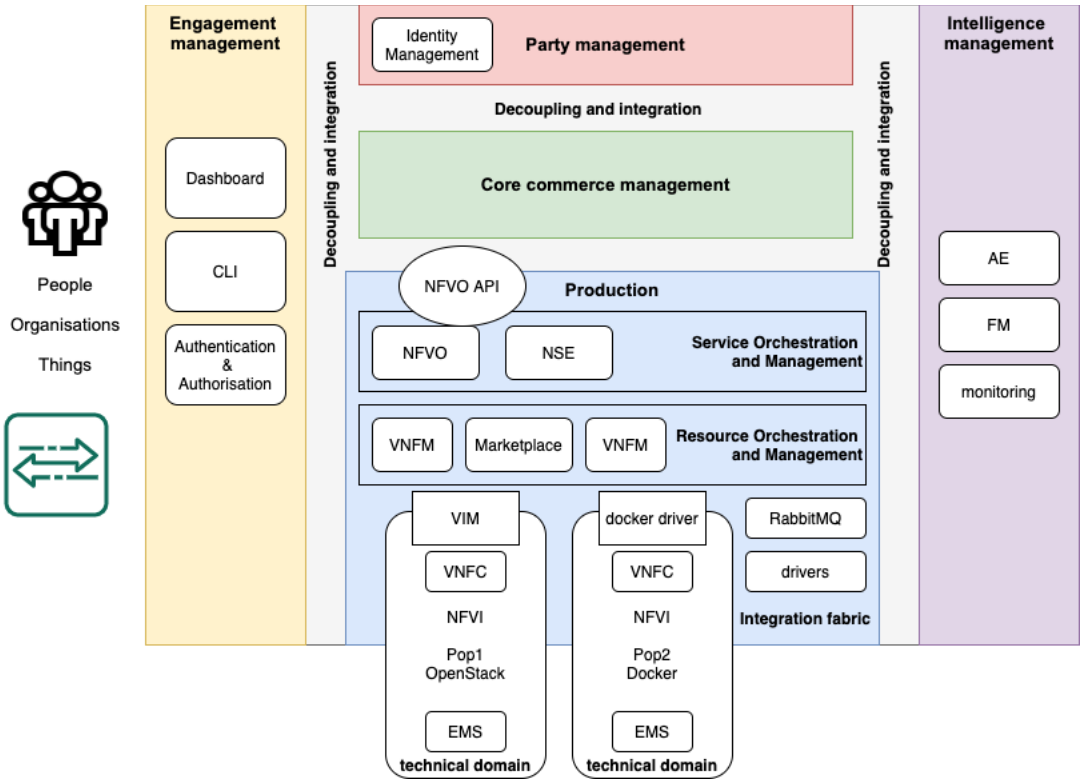

Figure B.14: Open Baton mapped to the ODA functional blocks

Most of the components belong to the Production block, as expected, while some can be mapped to other functional blocks to some extent, depending on their degree of disaggregation from the rest of the implementation.

Engagement Management

The user and admin interaction are provided via the Dashboard GUI or a CLI. No matter which type of interaction is selected, the authentication and authorisation mechanisms are applied first and only if that process is successful will the interaction continue. Both means of interaction are disaggregated from the rest of the components via the NFVO API. Thus, these components are placed in the Engagement Management functional block.

Party Management

Similarly, the Identity Management module, which manages the users, projects and roles, is mapped to the Party Management functional block.

Core Commerce Management

No obvious component candidates map to the Core Commerce Management functional block, since it is expected that the implementation should be integrated with higher-level system components that should provide these functionalities.

Production

The Production functional block contains most of the main features implemented in Open Baton. The service orchestration and management features are implemented via the NFVO, which closely follows

the ETSI MANO specification. There is a Network Slicing Engine (NSE), which is used for quality of service (QoS) management of network slices via JSON- or YAML-defined policies.

The resource orchestration and management functionalities are focused on VNF lifecycle management, which is implemented using a generic VNFM and an additional generic Element Management System (EMS) on the level of a VNF Controller (VNFC). The orchestration logic is completely separated from the VIM implementation via a VIM driver. Service function chaining (SFC) is orchestrated using the specialised Service Function Chaining Orchestrator (SFCO), which is tasked with mapping the defined SFC to optimal network paths. The NFVO orchestrates the lifecycle of the SFC instances in multiple points of presence (PoPs), while the SFCO translates the higher-level abstract request to specific SF chains with given QoS.

The integration fabric in the Production functional area consists of multiple special drivers and adapters that enable integration with different technological domains in addition to the default OpenStack-based NFVI. Using special adapters, Open Baton supports deployment of Juju charms and Docker containers. The aim of the drivers-based implementation is to enable interoperability with different solutions. In addition, all events are dispatched via the RabbitMQ using a publisher/subscriber model which supports flexible addition of new components and functionalities.

The network services are deployed as a combination of multiple VNFs using a Network Service Descriptor template file written in JSON or TOSCA [NSDT20] and based on the ETSI MANO specification. This NFVO template describes the VNF parameters and their connections via Virtual Links. The network service VNFs can then be onboarded in one or more PoPs using the Dashboard or CLI. The onboarding process creates a Network Service Record and launches the VMs in the PoP. The VIM is in charge of the NFVI resources management within one PoP. In other words, each PoP can be considered as a separate technical domain.

Intelligence Management

The implementation also includes autoscaling of operations options via the Autoscaling Engine (AE), where scaling policies can be defined using TOSCA YAML descriptors, and automated fault management (FM), both of which can be considered as implementations of special closed control loops and are thus located in the Intelligence Management functional block. There is also a monitoring plugin that provides integration with Zabbix, which can be used as a monitoring system.

The NFVO exposes an API that can be used by users and services with different methods of authentication and authorisation, where a service is considered to be any enabled entity that has the right to modify deployed resources. All communication between the components and the NFVO is done via this northbound interface exposed as a REST API.

B.1.6 Generalized Virtualization Model (GVM)

The Generalized Virtualization Model (GVM) describes an architecture that supports managing the lifecycle of different network topologies as virtual network slices over a physical infrastructure in a fully automated fashion. These virtual networks are environments where abstracted virtualised objects, called resources, can be defined, instantiated and arranged to create application-specific insulated networks and services. This process is performed via a graphical user interface (GUI) and

without the need for human intervention from network operations or engineering teams. The German NREN DFN has an implementation of this architecture in place called DFN-GVS (with GVS standing for Generalized Virtualization Service) [DFN20]. The work on this architecture ([SOB16], [HAZ14], [D8818]) began in 2013 based upon prior GÉANT work underway since 2010 (the GÉANT Open Flow Facility (GOFF)) [GOF14], and also incorporated concepts from a number of predecessor EU and international programmes such as FEDERICA [FED10], NOVI [NOV12], FIRE [FIR17], GENI [GEN18] and others.

As shown in Figure B.15, in GVM, users access the system via a User Agent such as a GUI and apply for a testbed or virtual network topology with all preferred resources using Domain Specific Language (DSL) code. A GVM-compliant service has a Provider Agent (PA), which is responsible for managing the infrastructure and the virtualisation services so that the requested resources can be instantiated. The Provider Agent consists of several components: the core Resource Manager (RM) or orchestrator, the Resource Control Agents (RCAs) and the Resource Database (RDB). The core Resource Manager sits behind the URL representing the GVM service. The RM interprets primitives received from User Agents and, after authenticating the UserID credentials, forwards the request to an appropriate Resource Control Agent for processing.

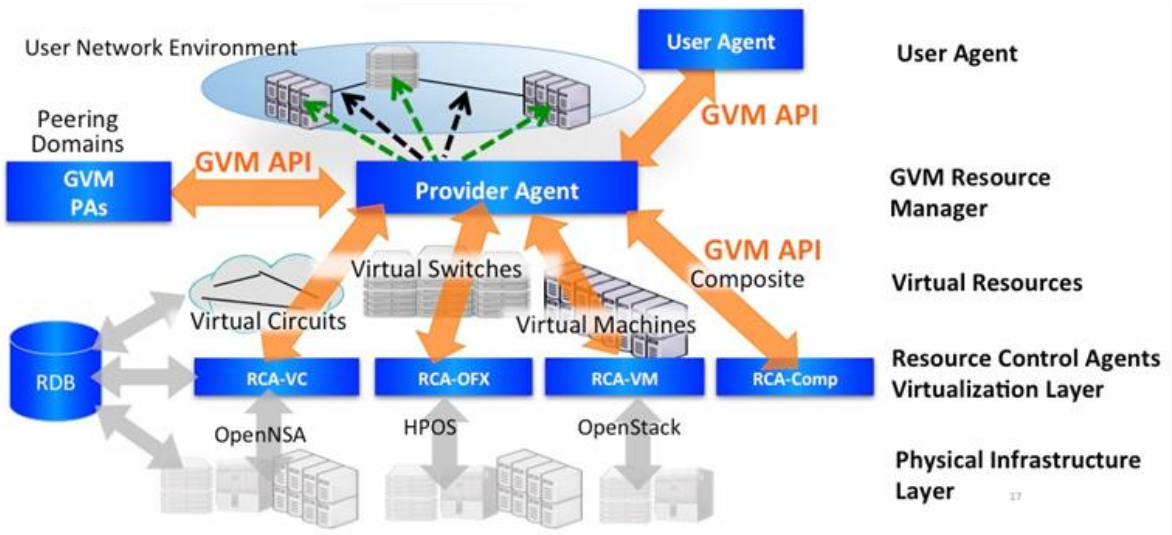

Figure B.15: Virtualisation, management and user control layers in GVM [D8818]

Each resource in GVM is a representation of various parts of the physical infrastructure. For each type of resource there is a Resource Control Agent that is responsible for instantiating these abstract resource classes in the physical infrastructure. A resource class has a synopsis attribute which describes its functional essence in a human-readable way. The class contains a set of parameters and primitives which are relevant for the certain type of physical resource. Every resource class implements five control primitives which move the resource through the states of its lifecycle:

- Activate() – go from RESERVED to ACTIVE state.
- Deactivate() – go from ACTIVE to RESERVED state.
- Reserve() – instantiate the resource, allocate physical resources.
- Release() – destroy the instance.
- Query() – get information about the actual state of the resource.

The Resource Control Agents implement these API primitives for each resource class and convert the GVM API to the infrastructure-specific actions needed to allocate, activate, deactivate, query and release each resource class. Thus, for each resource class there must be an RCA module that translates the generalised semantics of the GVM API to the appropriate infrastructure interactions. A well-designed RCA for a particular class should be able to support multiple types of underlying infrastructure by simply having different southbound interfaces for each type of hardware infrastructure. A user should only see the generalised resource model of the requested resource, but should not be exposed to the underlying mechanisms needed to deliver that resource to the user's environments.

To establish a resource instance, the RCAs create a resource record in the Resource Database, and allocate and reserve the infrastructure components that will be needed later to realise the resource in the infrastructure layer. At activation time, the RCA knows how to actually provision the allocated infrastructure components in order to construct the instance and place it in service. The Resource Database holds the authoritative state of the GVM-compliant service.

The work of GVM evolved in parallel with standardisation initiatives such as ETSI NFV and ETSI MANO, and they adhere to the same principles such as layering, abstraction/virtualisation and modularity [VAL19]. The GVM architecture can be mapped to ETSI NFV and MANO (Figure B.16).

The ETSI NFV Management and Orchestration functional block contains three elements: the Orchestrator, the VNF Manager and the Virtualised Infrastructure Manager (VIM). In GVM the Orchestrator and VNF Manager can be compared to the Resource Manager as a core orchestrator which runs the resources' RCAs that are provided for each specific resource or VNF. The Service, VNF and Infrastructure Description component is comparable to GVM's DSL code, which lists the resource attributes and network topologies [VAL19].

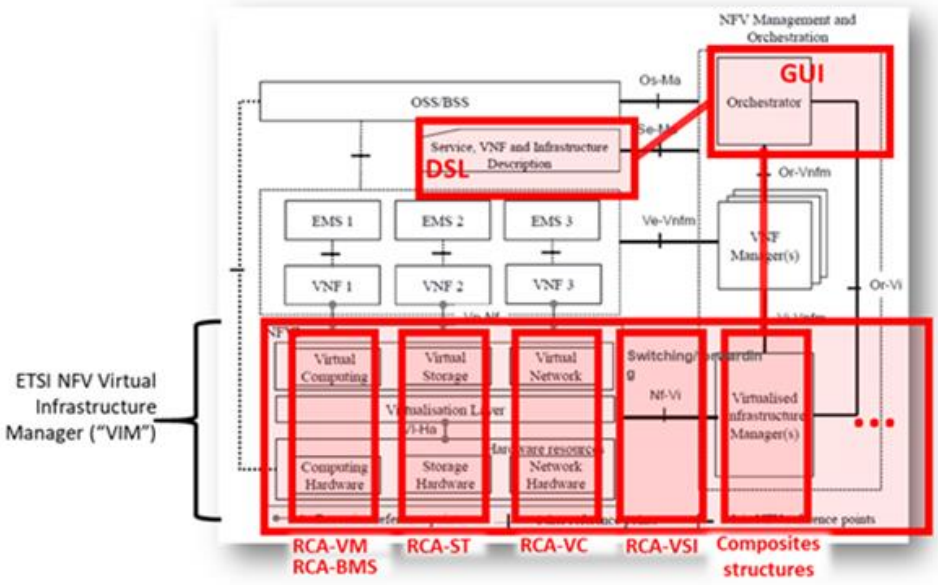

Figure B.16: Mapping of GVM and ETSI NFV architectural models

The authors of [VAL19] also compare GVM to ETSI-compliant implementations of the NFV MANO such as Open Source MANO (OSM) [EOSM20] and Open Baton [OB20]. They list similarities between the OSM Service Orchestrator (SO) and the GVM Resource Manager, compare the OSM Resource Orchestration (RO) to the OpenStack-specific RCA-OS in GVM and relate VNF Configuration and Abstraction (VCA) in OSM to the resource-specific RCAs in GVM. As far as Open Baton is concerned, the authors highlight similarities between Open Baton's NFVO, responsible for lifecycle management, and GVMs Resource Manager, point out resemblances between Open Baton's various drivers for allocation of resources and GVM's RCAs, and describe the users' request and associated package creations in Open Baton to GVM's DSL files.

The GVM architecture mapping with the TM Forum ODA functional blocks [ODAW18], [ODAFA21] is presented in Figure B.17.

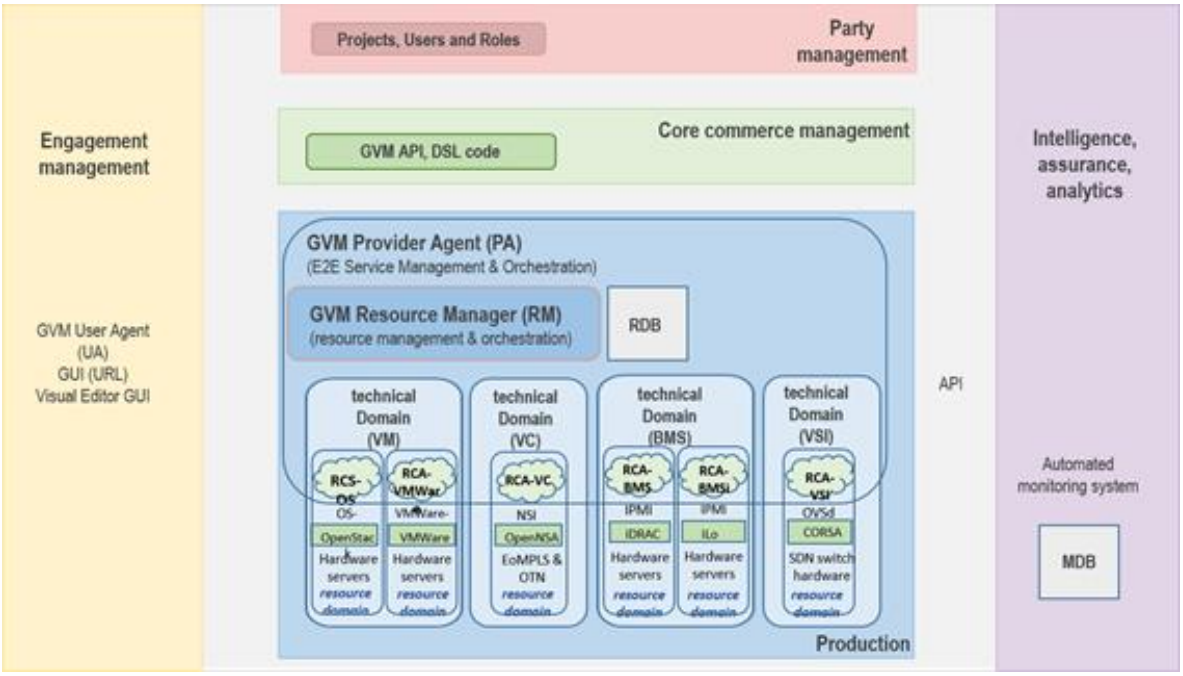

Figure B.17: GVM architecture mapped to the ODA functional blocks

Engagement Management

The Engagement Management domain includes the GVM User Agent, and GUIs. A user executes all control primitives via the GUI by submitting a DSL file. DSL code is based on Groovy (an object-oriented language for the Java platform [GRO18]) and allows advanced users to construct very complex and large networks in a quick and easy manner by applying iterations. For beginners, a visual editor offers a drag-and-drop GUI for easy creation of testbed environments. This visual editor allows users to pull down menus for resource selection and configuration, and enables the researcher to click on the resource objects and drag and connect them to other resources. While the researcher is designing their testbed using the mouse, the associated DSL is automatically being created in the background.

A RESTful API for application designers and system integrators allows reservations, activations, deactivations, releases and queries of resources via API calls. Therefore, user interactions with the GVM Resource Manager can be accomplished natively from user applications without the GUI. This

means that other possible user interfaces – or other orchestrators or GUIs - can interact with the GVM virtualised services to construct virtual networks.

Party Management

Each user is assigned the role of a "project owner" when first registering a project; each "project owner" can assign team members as "users" to his/her own registered project, but cannot grant the role of "administrator" to a "user". Only an "administrator" (i.e. a super user in charge of the GVM instance) can approve a "project owner". "Users" can edit their own user data, except for user role and project assignment. A user who registers a project must supply the user name, email address, name of the organisation, a login ID and a password (which the user can change at any time via the web portal). The user is also asked to provide start and end dates of a project and a short description, with estimated resource requirements, when registering a project. The information is stored in a single source of truth database. Further GVM processing and services are based on user, project and resource IDs.

Core Commerce Management

The Resource Database (RDB) holds the authoritative state of the GVM-compliant service. For each resource instance a record is created in the RDB and the corresponding Resource Control Agent (RCA) allocates and reserves the requested resource infrastructure components needed to realise the network service object.

Key information structures contained in the RDB are a Project Table, a Class Template Table, a Resource Tree, a Port Table and Infrastructure Tables. The Project Table delivers all information that defines a Project created within a service domain. Projects are equivalent to "accounts" and may contain a list of users authorised to reserve and manage resources attached to this project, a list of class templates defined under this project, and a list of resource instances that have been reserved. The Resource Tree (RT) represents all the resources instantiated in a single virtual testbed or network slice. Each RT is linked under a Project, and is constructed during the recursive reservation process. Intermediate nodes stand for composite resources in the tree, and each atomic resource instance is represented as a leaf node in the tree. A second database is used for monitoring. It keeps track of monitoring data of all resources and infrastructure components and offers tenant-based monitoring statistics of individual testbed environments.

Production

GVM implements a virtualisation layer as part of the Provider Agent, associating each resource with an RCA component that is capable of mapping that virtual resource object to the underlying infrastructure. The Resource Manager as the core orchestrator of GVM manages these RCAs and distributes these virtual resources. Once a user has submitted DSL code, the Resource Manager parses the DSL and calls up the respective RCAs for each requested resource type. Each RCA then creates a resource record in the RDB to reserve the resource and hands back resource IDs. At that point, all infrastructure components that will be needed later to realise the requested resource are reserved, but are not mapped to the infrastructure by the RCA until the resource is activated by the user. Only at activation time is the resource actually placed in service, i.e. the RCA modules translate these common primitives into the southbound hardware or infrastructure-specific command sequences necessary to have an effect on the function in other third-party codes.

Resources remain as records in the RDB until they are not only deactivated, but also released. Activation, deactivation and release are controlled by the user via the GUI and API primitives.

As an example, the resources that were available in the DFN-GVS instance of GVM as of version 6.1.5 were Virtual Machines (VMs), Virtual Circuits (VCs) and Virtual Switch Instances (VSIs), which offer a fully virtualised hardware instance of an OpenFlow switch. Additional resources can be added as long as the appropriate RCAs are made available.

Intelligence Management

Intelligence Management functionalities in the GVM architecture are foreseen in the design, but not yet implemented in their full potential. Monitoring should be set up to continuously observe and record parameter values associated with an object and collect historical data for comparison in order to be able to make judgements about the behaviour or state of the object as a whole. The monitoring of all resources should also include tenant-based monitoring of resources for statistics on users' testbeds and their resource consumption.

The monitoring processes must be directly linked to the lifecycle of resources, i.e. when a resource and its attributes are described in the DSL code, the associated monitoring for these resources and resource attributes must also be put in place. Furthermore, it should also be possible for a user to ask for certain metrics to be monitored via DSL attribute statements. On the other hand, there are also mandatory metrics that must automatically be instantiated to be monitored by the NOC when that resource type appears in a DSL file. All monitoring requests should be handled by the Query() primitive of each RCA. Once a resource is reserved through the Reserve() control primitive, an empty monitoring instance of that resource must be automatically created so that it can be filled with monitoring data as soon as the resource becomes activated.

B.1.7 SENSE

SDN for End-to-end Networked Science at the Exascale (SENSE) [SEN20], [MON18] was developed in a research project with the goal to provide a network architecture that will facilitate the rapid deployment of smart network services in support of science applications. The main contributors to this project are ESnet/Lawrence Berkeley National Laboratory, California Institute of Technology, Fermi National Accelerator Laboratory, Argonne National Laboratory and the University of Maryland College Park.

The architecture was developed with next-generation big data/exascale science workflows in mind that will require intelligent network services in cloud computing, AI and machine learning. The idea is to support scientists and their science workflows with a system that will automatically build virtual guaranteed networks over large-scale distributed storage and computing infrastructures without any human intervention and without the static non-interactive network infrastructures in place today.

SENSE is planned to be deployed in a realm of FABRIC [FAB19], the new research infrastructure across the US which is funded by the National Science Foundation (NSF) in an effort to support innovative next-generation applications and computer networking. Another use case for SENSE is being prototyped together with the AutoGOLE community of networks that support the Network Service Interface (NSI) as a means for network provisioning [MAL20]. In this case, SENSE is envisaged for the

provisioning of multi-resource services including networks and data transfer nodes (DTNs). The integrated SENSE/AutoGOLE work is pursued within the Global Network Advancement Group (GNA-G) collaboration [GNA-G].

The development of SENSE was driven by the following design principles and requirements:

- End-to-end, for multi-domain networks, including their end systems and network stacks.
- Real-time resource provisioning and availability (scale seconds to minutes).
- Intent-based interfaces that will allow applications to define high-level service requirements that will then be translated into the intended network services by an orchestrator.
- Interactive, so that applications can interact with networks to receive status information or negotiate performance parameters.
- Full service lifecycle interactions, between network and application, for a continuous exchange during the lifetime of a service.

The SENSE architecture (see Figure B.18) describes an intelligent network service plane with an underlying hierarchical service-resource architecture where the two major functional roles are SENSE-O orchestrators and SENSE resource managers.

The SENSE orchestrator accepts requests from the user application and determines the appropriate resource manager that could fulfil that request. An orchestrator is typically associated with a domain science application (for example, Large Hadron Collider / Compact Muon Solenoid (LHC/CMS) [LHC20]) and would be able to process high-level context-sensitive requests and turn them over into low-level descriptive resource requests.

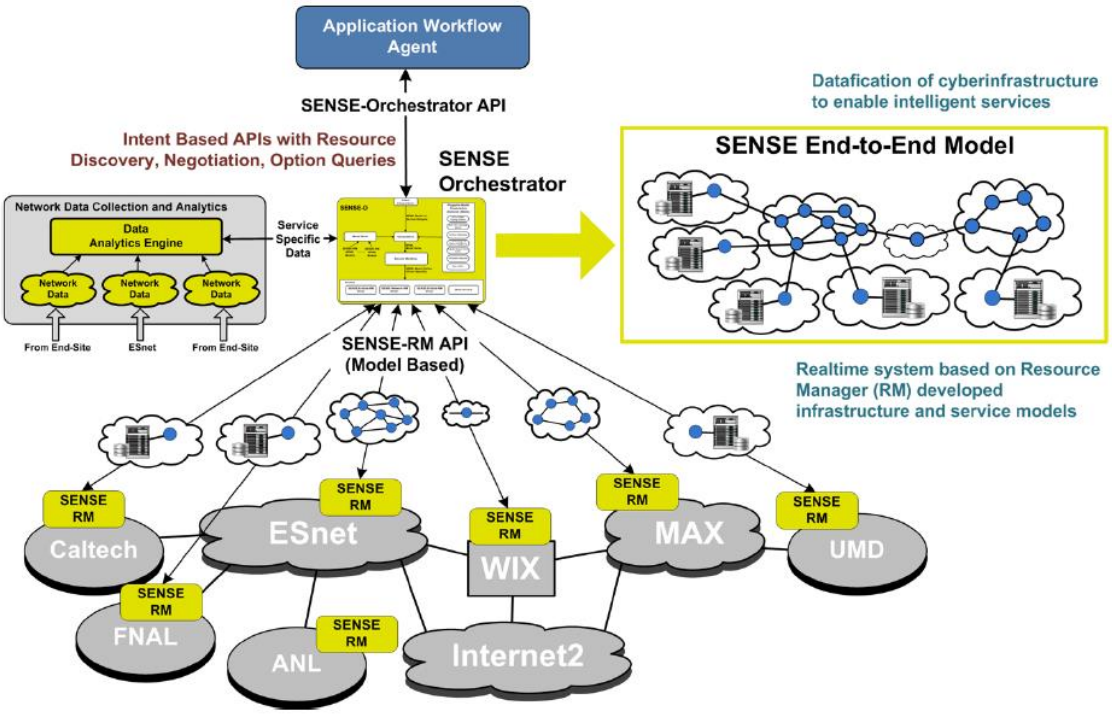

Figure B.18: SENSE architecture [MON18]

However, SENSE-O is not a singular central orchestrator; rather, the architecture provides many instances of orchestrators where each orchestrator instance serves a different organisation or application and also deals with multiple SENSE resource managers, depending on the requirements that each application workflow/service request may have. This SENSE orchestration model, with its many-to-many relationship between orchestrators and resource managers, makes it possible to associate each orchestrator instance with its own access and identity framework and specific application policies and service-level agreements (SLAs), as needed for that service workflow. At the same time, the relationship between an orchestrator instance and its resource manager instances also allows fine-grained access to resources and can provide resource restrictions to one resource manager instance that should not apply to other instances.

Figure B.19 provides a summary overview of how the SENSE architectural components described above can be mapped to ODA functional blocks.

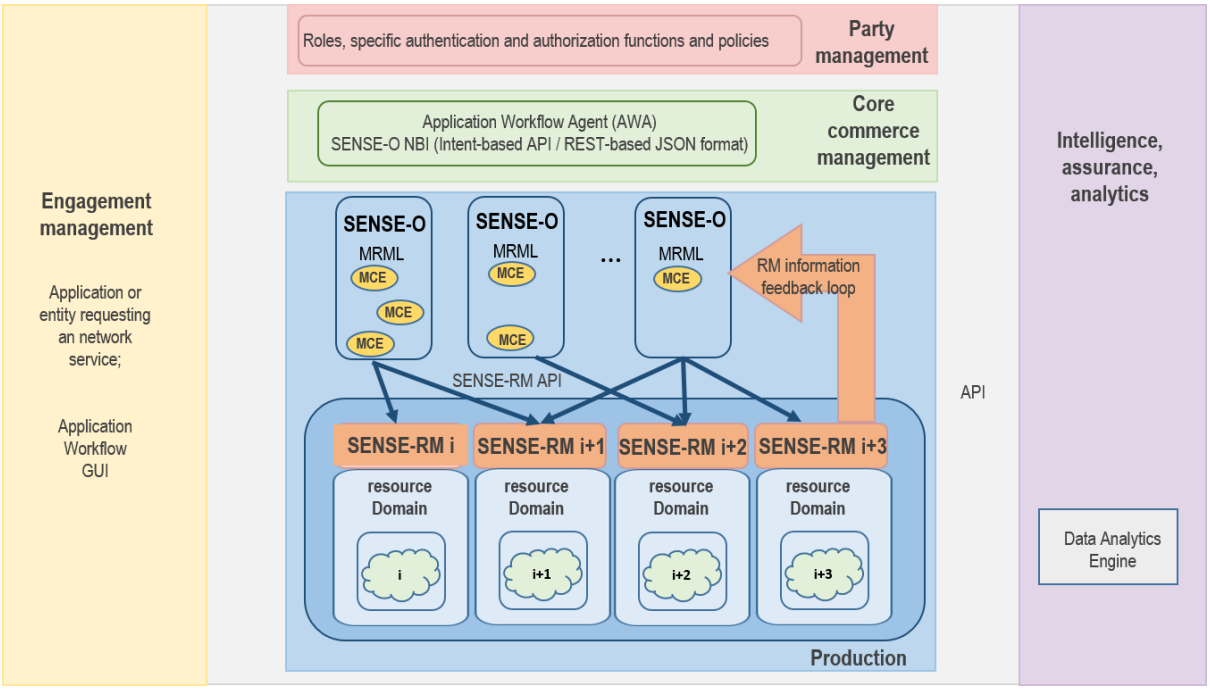

Figure B.19: SENSE architectural components mapped to the ODA functional blocks

Engagement Management

In SENSE, smart applications are intended to interact with the smart network in the intelligent network service plane. An application or entity requesting a network service can engage with the system via an Application Workflow GUI.

Party Management

The SENSE architecture was specifically developed for scientific applications where customers (or parties) need network services for big data transfers and may have stringent bandwidth or quality of service (QoS) requests. For this reason, the orchestration model of SENSE assigns different instances of SENSE-O orchestrators to different application workflows so that each service request can be associated with specific authentication and authorisation functions and policies. Therefore, end-user

access can be limited to resources within that orchestration and associated resource domain, and there is no need for the management of all end users by all orchestrator instances.

Core Commerce Management

An application or entity requesting a network service is called an Application Workflow Agent (AWA) and is considered a smart application component that is capable of engaging with the smart network to obtain services from there. This service negotiation workflow is conducted via an intent-based API referred to as SENSE-O NBI, a northbound programmable interface that offers application workflow assistance. SENSE-O NBI provides a set of other intent-based API calls, which can handle such service negotiations, resource discovery, intelligent bandwidth and network computation and workflow instantiation. The SENSE-O NBI service intent is a REST-based API using JSON format [NBI20]. For the user, the SENSE orchestrator (instance) arranges the application service through this SENSE-O NBI by providing a workflow instantiation comprising service creation, negotiation, modification, cancellation and service monitoring calls.

For the core of SENSE-O, StackV [STA20] is implemented, which is a general-purpose open source orchestrator. The SENSE-O NBI translates the abstract service intent coming from the application into a Service Model Description and Abstraction, which is a Multi-Resource Markup Language (MRML) description [MRM20]. MRMLs are extensions to the Network Markup Language (NML) standard by the Open Grid Forum [HAM13], [OGF20] which include additional resource types aside from the basic network and topology elements, such as Data Transfer Nodes (DTNs), storage systems, compute nodes and scientific instruments.

Production

The SENSE-O orchestrator instance polls several resource managers (RMs) for their resource models, which describe what each RM has to offer as far as its local resources are concerned and also list how each resource manager's resources are interconnected to external resources. With this information, the SENSE-O orchestrator is able to construct a connected graph with all required RMs and can ultimately turn this graph into workflows for actual resource provisioning. In other words, the SENSE-RM API offers a mechanism to integrate separate resource domains under the orchestration domain and supports the many-to-many relationship between SENSE-RMs and SENSE-Os as described above.

Once the orchestrator computes the system delta that leads to the requested topology, it suggests this delta to the corresponding RM, which can then either accept or reject it. Rejection is possible, for example, if resource requests cannot be fulfilled or local usage policies prevent fulfilment. Once the delta is accepted by the RM, the orchestrator designates the change in a final bind and the RM will be able to carry out the change to the corresponding resources.

The MRML documents (models) that RMs provide to describe their resources and topology information are constantly updated by the SENSE-RMs (for example, every 30 seconds) in order to have real-time information available for the SENSE-O that reflects the current resource situation in each RM's domain. If an MRML document is modified, the SENSE-O orchestrator then pulls this new data and thus has a recent model description for use in its multi-domain resource description model. With every change in resources or topology, the SENSE-RM needs to maintain a new version of this MRML document.

For each service request and calculated system delta, the SENSE-O first propagates the delta to the SENSE-RM(s). This propagated delta message leads the RMs to verify and confirm the request, but it

takes an additional commit delta message by the SENSE-O to the corresponding SENSE-RMs to actually provision the resources. The propagated delta message also contains an element for negotiation in case updated real-time synchronisation on the status of resources is needed.

Intelligence Management

The SENSE architecture also includes a Data Analytics Engine which is external to the orchestrator and collects network telemetry and resource utilisation data. It is not yet implemented in the current SENSE prototype, but will be an important component towards making additional network analytical data, historical network measurements and feedback information available for improved descriptions of service intents. Once implemented, this Data Analytics Engine will also be based on service-specific data models and will be integrated into the existing MRML model. Model computation elements (MCEs) will then be able to use this information also, for improved service function computations. In addition, the intention is to provide feedback from the SENSE-O to the Data Analytics Engine for a two-way feedback mechanism and a means to verify data collection.

B.1.8 Open Network Automation Platform

The Open Network Automation Platform (ONAP) [ONAP20] arose after the combination of two Network Function Virtualisation (NFV) orchestration and management initiatives: Enhanced Control, Orchestration, Management & Policy (ECOMP), led by AT&T, and Open Orchestrator (Open-O), led by China Mobile.

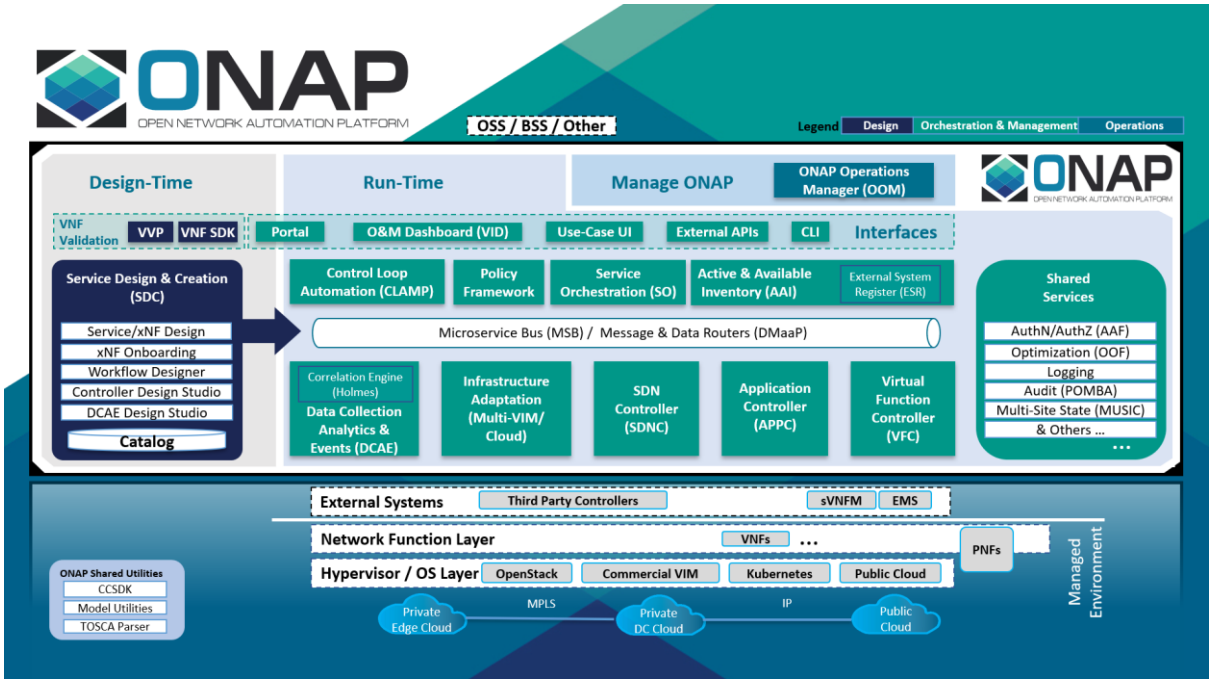

Figure B.20: ONAP architecture [ONAP20]

ONAP provides an automation platform for virtual and physical network elements, enabling a rapid deployment of network services. ONAP constitutes a platform on top of the infrastructure layer which

provides the end users with tools for the creation, orchestration, automation and handling of the full lifecycle management of Virtual Network Functions (VNFs), Software-Defined Networks (SDNs) and the end-to-end services built through them.

The ONAP platform is partially aligned with the ETSI NFV architecture. It supports the VNFD and NSD information models and supports functionality defined by ETSI MANO, including the NFVO interfaces.

ONAP has a flexible microservices design in order to suit different scenarios and environments. The ONAP components are deployed as docker containers with Kubernetes. The ONAP architecture consists of three types of functions: design-time functions, run-time functions, and functions for managing ONAP itself (ONAP Operations Manager), as depicted in Figure B.20. It also has external APIs for northbound interoperability and supports the integration with multiple VIM, VNFM and SDNC technologies, as well as legacy equipment.

The ONAP Operations Manager (OOM) takes care of the lifecycle management of the ONAP platform. It is responsible for the deployment and configuration of the ONAP platform. It performs real-time health monitoring of the components, and scales, upgrades or deletes ONAP components, including automatically restarting failed ONAP components.

The design-time framework provides a development environment that makes it possible to describe the resources, services and products. It consists of two subsystems: the Service Design & Creation (SDC) subsystem and the Policy subsystem.

- The SDC allows product and service designers to perform operations on the resources and services, such as onboarding, extending or retiring them. The SDC consists of a Catalogue, which is the repository for resources and services; a Design Studio, to create and modify resources and services from the Catalogue; a Certification Studio, to perform testing on new assets; and a Distribution Studio, which is used to deploy certified assets.
- The Policy subsystem provides an environment for the creation and management of policies. Policies are rules that affect the behaviour of components, which can be infrastructure, products or services. Policies specify conditions and requirements that need to be satisfied.

The run-time framework is responsible for the execution of the policies that have been designed within the design-time framework. It consists of several components:

- The Service Orchestrator (SO) provides a high-level view of the infrastructure and high-level orchestration of services and resources, including Virtual Network Functions (VNFs), Container Network Functions (CNFs) and Physical Network Functions (PNFs).
- The Virtual Infrastructure Deployment (VID) enables the instantiation of services from the SDC, and lifecycle operations to be performed on them, such as scaling and upgrading VNFs.
- The Active and Available Inventory (A&AI) provides real-time views of the system's resources, services, products and their relationships with each other.
- The Policy Framework is the decision-making component in ONAP, supporting multiple policy engines.
- Different types of controllers, including SDN Controller (SDNC), Application Controller (APPC) and Virtual Function Controller (VFC), can be included in the system to manage different types of resources within the domain of each controller. For example, the Virtual Function Controller provides an ETSI NFV-compliant NFVO function for the management of virtual services.

- The Multi-VIM/Cloud Infrastructure Adaptation enables ONAP to adapt to multiple infrastructure environments, such as OpenStack.

The mapping of ONAP to ODA is presented in Figure B.21.

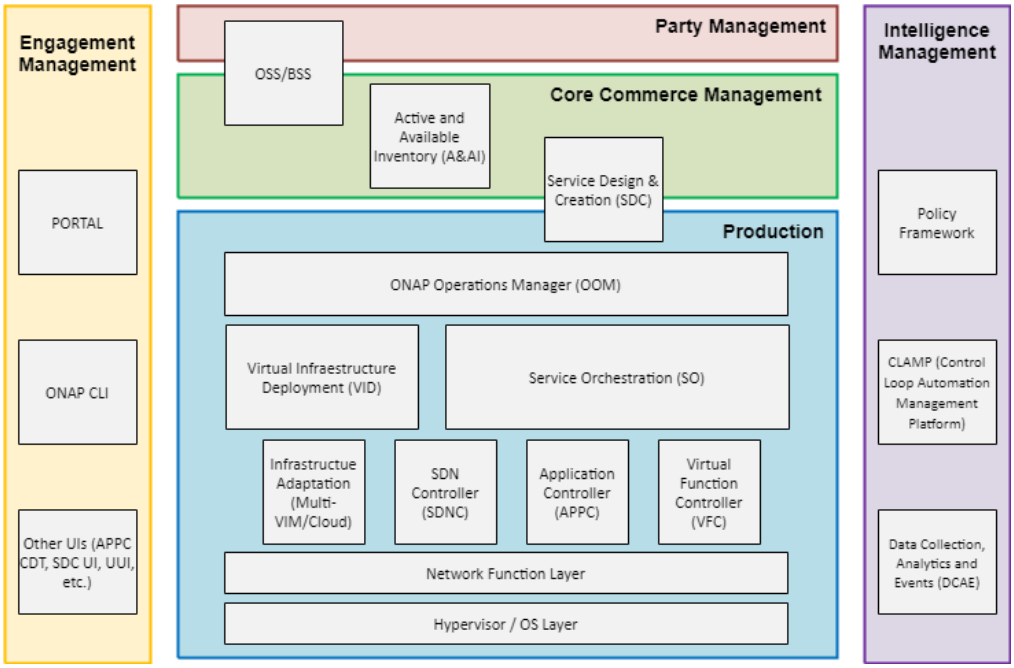

Figure B.21: ONAP mapped to the ODA functional blocks

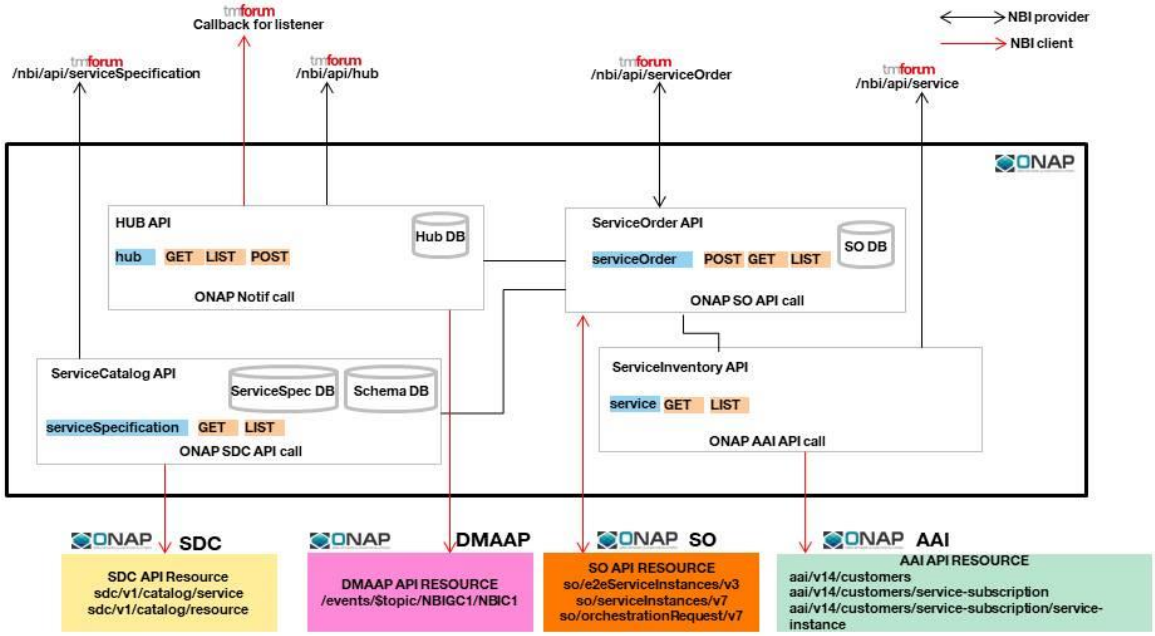

Figure B.22: NBI architecture and its integration with other ONAP components [ONAP20]

Engagement Management

ONAP provides a Northbound Interface (NBI) based on a Java 8 web application. The interface is built over the Spring Framework and it runs as a standalone application with an embedded Tomcat server. Figure B.22 presents the NBI architecture and its integration with other ONAP components (SO, SDC, etc.), as well as the API resources provided. The NBI offers the following five APIs: hub, serviceCatalog, serviceInventory, serviceOrder and status.

Hub provides the ability to subscribe to and unsubscribe from external API notifications. The serviceCatalog API is based on TMF633 serviceCatalog and provides high-level information retrieved from the SDC. The serviceInventory API is based on TMF638 serviceInventory and provides high-level information about instantiated services that is retrieved from the A&AI inventory. The serviceOrder is used to request the instantiation of a service. A Service Order, triggered by a BSS system, is a type of order that can be used to describe a group of operations on a service (e.g. add or terminate). The status API provides a health check of the NBI component.

The ONAP platform has a set of user interfaces (UIs) to command the system in a more user-friendly way. The available user interfaces are:

- Application Controller Controller Design Tool (APPC CDT). This tool provides a graphical user interface (GUI) for the onboarding and management of VNFs in ONAP.
- Control Loop Automation Management Platform (CLAMP) Dashboard. A Kibana-based GUI that allows users to visualise control loop metrics through a dashboard.
- Command Line Interface (CLI). The CLI can be used both in scripting mode and interactive mode. It is available when accessing the server where CLI is installed, or via the web console in port 30260.
- Portal. The ONAP Portal provides a user interface to both design-time and run-time environments, based on the user's role. It provides access to design, analytics and operational control and administration functions via a dashboard, enabling application onboarding and management.
- Service Design and Creation User Interface (SDC UI). The request to the SDC can be submitted by means of a Kibana-based GUI, and is then passed to a Jetty front-end server for its execution. The Kibana server enables statistical analysis of the operations according to the business logic.
- Use Case User Interface (UUI). The UUI module provides a GUI for operators and end users. It displays the service catalogue and provides the means for NS/VNF onboarding. It also shows alarm and performance data.
- Virtual Infrastructure Deployment User Interface (VID UI). The VID UI provides the means to invoke the instantiation of a service and all of its subcomponents (e.g. VNF/VF, VNFC/VFC) as well as cloud logical environments.
- Active and Available Inventory Graphical User Interface (A&AI GUI). This UI is a tool that allows end users to obtain graphical information about the resources available in the A&AI. This UI is deployed in two parts: a back-end component and a front-end component.

Party Management

In ONAP, the Portal enables administrators to manage ONAP users and roles, for example, to create or delete users, or assign roles to users to configure their privileges with regard to different ONAP applications such as the SDC, the A&AI and the VID (examples of roles could be "Admin", "Designer",

or “Tester” for the SDC). ONAP also provides the External API Northbound Interface, providing an abstracted view of the platform to external OSS/BSS systems. Third-party users can make use of ONAP through an OSS/BSS system integrated with ONAP.

Core Commerce Management

A service in ONAP is designed and created through the design-time framework, and in particular the Service Design & Creation (SDC) component. The design-time framework allows the specification and modelling of all the elements of a service. A service includes not only the different kinds of resources that make it up, but also the policy rules that determine the behaviour of the service, as well as the control loop events that enable the monitoring and lifecycle management of the service in an automated way.

The ONAP community has developed a set of blueprints [ONAPB20] in order to illustrate real-world use cases of ONAP. These blueprints are aimed at use cases that are a priority for the ONAP community. Examples include the 5G blueprint, the Virtual Customer Premises Equipment (vCPE) blueprint, the Voice over Long-Term Evolution (VoLTE) blueprint, and the virtual firewall/virtual DNS blueprint.

The inventory management in ONAP is realised through the Active and Available Inventory (A&AI) subsystem. The A&AI provides real-time and historical views of the system’s resources, services, products and their relationships. A&AI can relate data from multiple ONAP instances, BSS/OSS systems and applications. It maintains a registry that ranges from the end products to the low-level resources that make up the products. In addition, A&AI dynamically receives automatic updates of the inventory as controllers such as an SDNC make changes in the environment.

Production

ONAP components involved in Production management are presented in Figure B.21 above (OOM, VID, SO, Multi-VIM/Cloud Infrastructure Adaptation, controllers, etc.) and were described in the ONAP overview (see pp. 42–43). This section explains the onboarding process in ONAP that is required in order to offer services.

To create a service, the service first needs to be designed and the service package onboarded. Each service requires a tenant in the cloud infrastructure outside of ONAP, where the resources of the service are to be deployed. This tenant has the virtualised computing and networking resources required to deploy VNFs. The process of registering a tenant is done through the A&AI component.

Services can be created through the SDC component. Two types of resources are supported: Physical Network Functions (PNFs) and Virtual Functions (VFs). After a VNF has been validated and onboarded, it is added to the SDC Catalogue. VFs and PNFs can be created either manually, or by importing a Virtual Software Product (VSP). VF/PNF information also includes deployment artifacts (for example, a Heat file), information artifacts (such as test scripts), and TOSCA artifacts. Finally, the uploaded VF/PNF is submitted for testing. Existing VFs/PNFs can be updated or deleted through the SDC Catalogue in the GUI.

Intelligence Management

The two primary components concerned with the automation and monitoring tasks in ONAP are the Data Collection, Analytics and Events (DCAE), and the Control Loop Automation Management Platform

(CLAMP). CLAMP enables the automation of functions related to the lifecycle management of VNFs and VMs, as well as of the ONAP components themselves. A control loop can be configured with parameters that satisfy the lifecycle management requirements of a specific network service, and updated at runtime after the network service is deployed.

The Data Collection, Analytics and Events (DCAE) is a component from the run-time framework which provides the closed control loop automation. It collects performance, usage and configuration data from the virtual network functions and their underlying infrastructure. The data collected by the DCAE Collection Framework can be sent to other components, such as controllers or the Policy Framework. These other components will trigger the corresponding actions, such as a fault event requiring healing or a capacity issue requiring scaling. DCAE also has an Analytics Framework, which enables the development of real-time and non-real-time analytic applications to process the collected data. These applications can be built for multiple purposes, including fault detection and troubleshooting, performance visualisation, capacity planning and security.

B.1.9 EOSC Architecture

The European Open Science Cloud (EOSC) initiative [EOS20] is part of the European Commission's effort to increase the competitive digital economy in Europe. The main idea of the initiative is to support Open Science, which in turn will enable a more agile approach to innovation. The main goal of EOSC is to provide researchers from different research fields with the means necessary not only to conduct research, but also to make their research outcomes open to the wide community. Thus, in essence, the end goal is to create a cloud-based virtual environment that will provide a number of open services that can be used to store, manage and analyse research data. Six different actions are being implemented in parallel to achieve this goal, described in the EOSC implementation roadmap [ETS20]. One of the action lines focuses on the architecture of the virtual environment, which should achieve a federated, seamless view of all available research infrastructures and services built on top.

The activities related to the definition and implementation of the EOSC architecture include:

- Organisation and implementation of the EOSC federating core which will facilitate the access to the federated infrastructures and monitor and regulate their use.
- Survey of existing data infrastructures and their ability to join the federation.
- Definition of the rules and requirements so that EOSC components and infrastructures can be added to the federation.
- Maintenance of a registry of federated data infrastructures.

The EOSC federating core aims to provide a single access point to all federated shared resources and a catalogue of EOSC products with underlying services that are provided by the federated infrastructures. The term "EOSC services" in this architecture is an equivalent to products from the ODA Core Commerce Management functional block. The EOSC services are divided into two main categories: generic services, such as data storage or general-purpose computing; and thematic services, which are discipline-specific research-oriented services. The single point of access to all data and services is to be via the EOSC Portal, which will also provide onboarding process activities for new data and service providers that would like to join. A separate activity on rules of participation will define the rules under which a provider can join and a user can use the EOSC services.

The main goal of the Architecture working group is to define the technical framework to support the EOSC vision. This framework includes standards, protocols and APIs needed to create an interoperable service environment of all federated providers. The components of the framework that need to be managed by EOSC are called the federating core. The work of the Architecture group is ongoing, but it is intended to consolidate development of the EOSCpilot and EOSC-hub projects [ETS20], [HTA20].

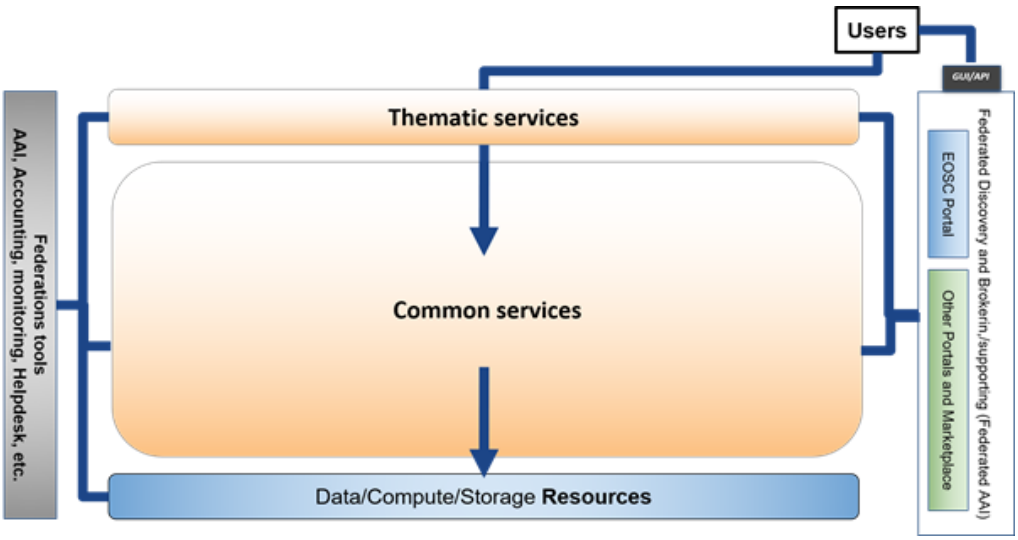

Figure B.23: EOSC technical architecture – functional view [HTA20]

The main features of the EOSC technical architecture should include service composability and interoperability [HTA20] (see Figure B.23). Thus, there should be the ability to combine several EOSC services into workflows (composite products) to make the service usage easier for researchers. The basic service features should be implemented by the underlying infrastructures, exposing only the relevant features for researchers (Figure B.24). To be able to successfully compose services end-to-end (as part of a product), the architecture must also support interoperability by adopting standards and well-known APIs. For these purposes it is proposed that the services are implemented using loose coupling of building blocks. The building blocks of the EOSC technical architecture are defined with their scope, features, policies, standards, APIs, etc. There are no fixed proposals for the implementation of each building block, only guidelines on how to implement EOSC-compliant building blocks. The definition of the API for infrastructure resource and common services is provided in [HTA20].

The loose coupling of building blocks and the definition of the federating core components on top of the federated infrastructure follow the OAV core design principles discussed in Part A of this document. The EOSC projects are also looking into similar standards when defining the building blocks' features, such as TM Forum Open APIs, TOSCA, OAuth, SAML, REST, etc. The concept of virtualisation is implemented within the federated infrastructures in order to provide multi-tenancy, and several layers of orchestration are present: within each infrastructure and at the level of EOSC federation, completed by the orchestration of researchers' workflows on the top. Automation is also implemented wherever possible, so that researchers can focus on using the services without the manual intervention of the service providers.

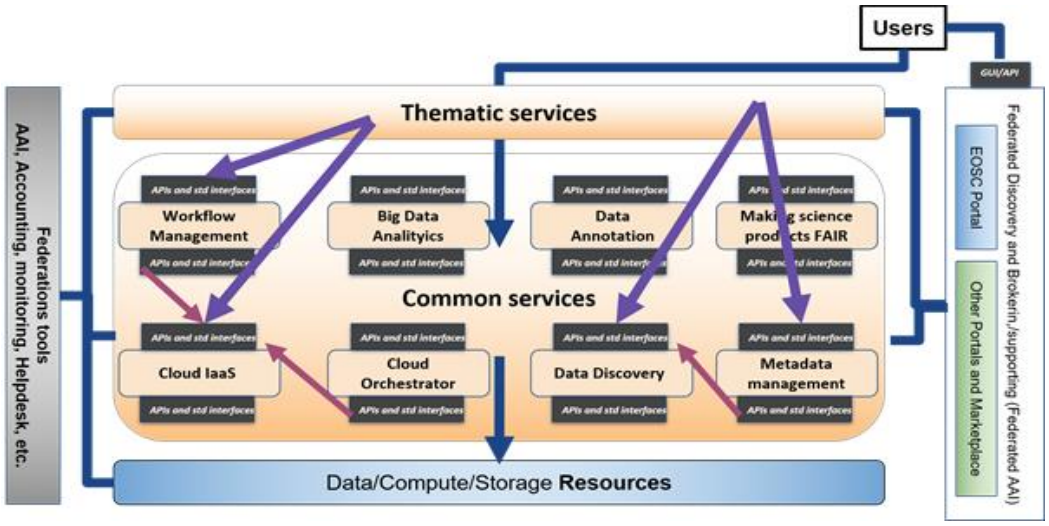

Figure B.24: EOSC technical architecture – interactions between thematic and common services [HTA20]

Figure B.25 below provides an ODA view of the current status of the EOSC architecture based on the outcomes of the EOSC-hub project.

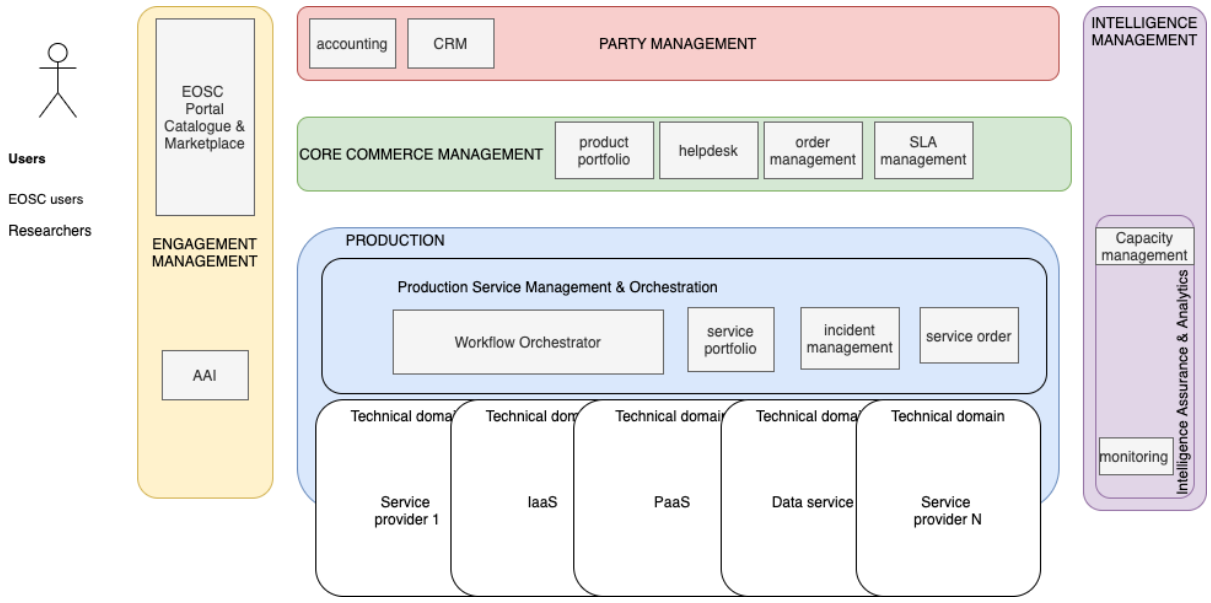

Figure B.25: EOSC architecture mapped to the ODA functional blocks

Engagement Management

The EOSC Portal Catalogue & Marketplace is considered to be the web GUI that will enable researchers to browse and order the available EOSC products. The Catalogue available on the Portal will enable researchers to view what products are available and under what rules and conditions, while the Marketplace will promote the use of highly mature products ready for use by the researchers. An AAI component will enable authentication and authorisation of the users. The current version of the EOSC Portal is available at [EOSCP].

Party Management

The roles and permissions that can vary for different users and sets of services are defined in the Party Management functional block in the form of a customer relationship management (CRM) component. In addition to the standard customer management, one of the main components of the federating core is the accounting component. The goal of this component is to track the use of products and potentially integrate with billing systems, depending on the way the products and underlying services are offered to the researchers. It is expected that some of the services will be free for use, while others will be paid for by researchers, research projects or other entities.

Core Commerce Management

The main catalogue presented in the EOSC Portal is a subset of all available products ("services" in EOSC terminology) for researchers, which are tracked in the product portfolio. The portfolio includes additional products, such as those that are planned, retired or under maintenance. Once the onboarding process is successfully completed, the products that a service provider wants to offer to the researchers are added to the product portfolio. The ordering of the products by the researchers is handled by the product ordering component, while SLA conformance is handled by the SLA management component. SLA conformance is defined based on SLA agreements for different products and the results gathered from monitoring. An overarching helpdesk is also part of the federating core, ensuring that a first line of support is available for the researchers. The second line of support can also be implemented in the federating core or can be forwarded to the service provider that provides the service in question.

Production

The main goal of the Production functional block is to provide an extra layer over the service providers and technologies so that interoperability and service composability are enabled. For these purposes, a common set of APIs and standards is envisaged to be used to interact with the different service providers, where each service provider is seen as a separate technical domain. If a service provider offers services of several different natures in different ways, then these could potentially be described in different technical domains as well. Also, within the technical domain there could be a hierarchy of hidden layers of orchestration and virtualisation, based on the implementation of the infrastructure. From a high-level view, each technical domain in ODA can be mapped to a separate infrastructure that wants to be federated into EOSC. The service management and orchestration layer provides the ability to compose different services. For these purposes there is also a service portfolio and service order management component. Each separate service is monitored, and if there are any problems these are reported and managed by the incident management component, which is also related to the helpdesk.

Intelligence Management

The monitoring repository can be seen as an Intelligence Management component, although at this stage there is no clear information on whether any type of closed automation loop will be implemented within the EOSC federating core. The information from the monitoring component may be passed to incident management in the event of a problem, and possibly to the accounting component for accounting purposes, although it is envisaged that the tracking of resource usage will be done via separate probes, distinct from monitoring, which aims only to check the service health via blackbox testing. The capacity management component is also being mentioned as a means to track the usage of the resources in the infrastructure and identify any bottlenecks or possibilities for load balancing.

B.1.10 ETSI GANA

One of the main requirements for future architectures and digital service providers is to be able to transform all manual management of network and services into Automated and Autonomic Management & Control (AMC) so that the system dynamically responds to the changing environment. This behaviour is also known as autonomics and is the main focus of the Generic Autonomic Network Architecture (GANA) [GANA16]. In other words, the GANA reference model provides a way to build autonomic architecture using a generic AMC framework that can integrate with any type of network management architecture.

The ETSI Network Technologies (NTECH) Autonomic network engineering for the self-managing Future Internet (AFI) working group leads the AMC standardisation efforts that will transform the traditional networking environment into a smart and intelligent network with a number of self-* features including self-healing, self-configuration, self-optimisation and others. The main work done by this working group is represented in the GANA reference model and implementation guide [AFITS18], together with the efforts to implement GANA into service/network management architectures defined by other standardisation organisations, with the 3rd Generation Partnership Project (3GPP) and the 5G architecture as one of the major use cases.

The main concepts behind AMC involve the implementation of a closed control loop via autonomic functions that are implemented using Decision-making Elements (DEs). The DEs include cognitive functions to learn and reason how to optimise and adapt to changes in the high-level business policy or a problem in the network that will affect the QoS or availability. The DEs' decisions are effected by controlling the behaviour of the associated Managed Entities (MEs) (see Figure B.26).

There are two general types of closed control loops: fast control loops, where the reaction to the change is in real time or near real time, and slow control loops, where the reaction is slower and usually involves a collaborative decision-making process. Fast control loops are implemented using a distributed approach, with DEs being deployed within the network elements, while slow control loops use a centralised architectural model, where the DEs are centralised into one location, removed from the network elements. In both cases, it is possible for a DE to exchange information with other DEs and thus implement a collaborative effort to reach the required global network state.

To be able to learn about the environment, DEs require information about the status of the network and services. Therefore, this architecture relies heavily on the ability to gather network analytics, both reactive and proactive, in real time. This information can be used to find patterns and make predictions about the network's future behaviour. In addition, the architecture also requires that the desired global network state is expressed using dynamic network and service policies which will be used as desired outputs that will guide the DEs' cognitive functions. It is natural that such an architecture can only be implemented in cases where the orchestration processes involving service provisioning and service assurance can be implemented on demand in an automated fashion, allowing changes to be made to the service configuration on the fly. It is thus important to distinguish between automation and autonomics: automation entails automation of workflows and processes that replace manual network configuration and monitoring, while autonomics involves cognitive functions that enable automated reaction to changes based on internal reasoning processes.

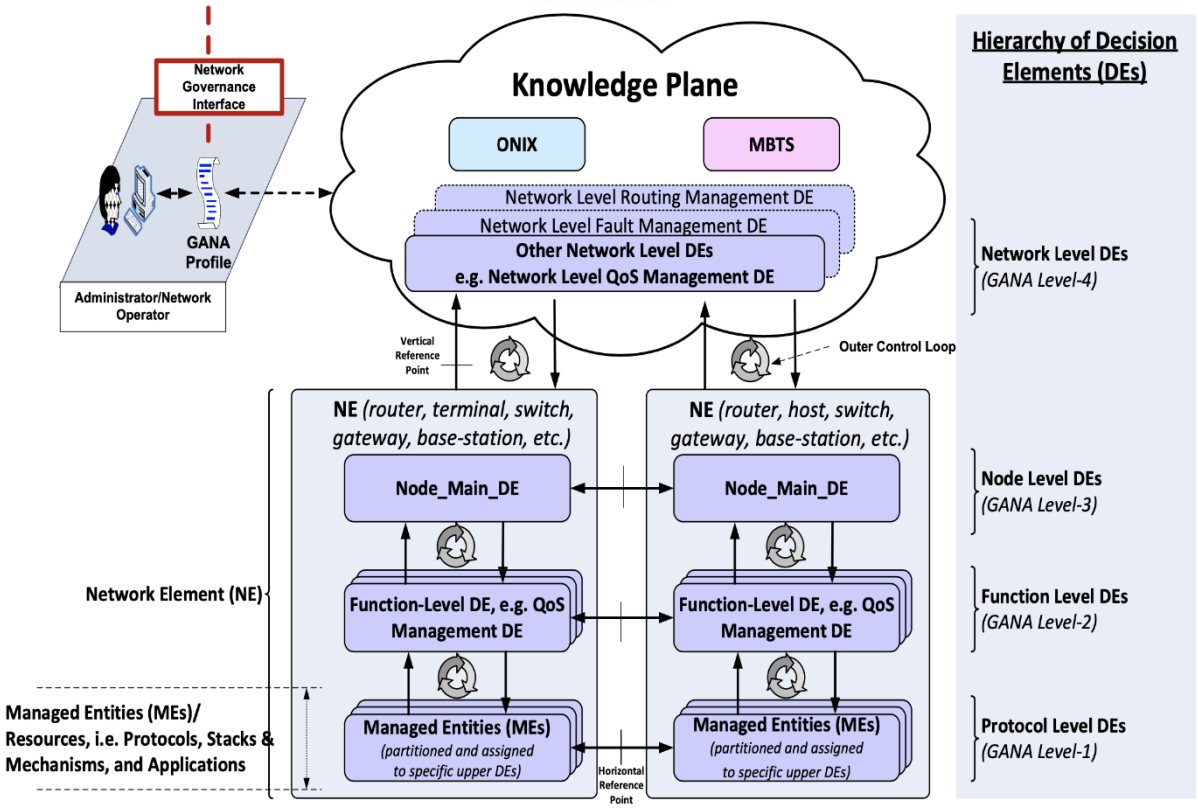

Figure B.26: AFI GANA reference model, taken from [GANA16]

At the top of Figure B.26 is the GANA Knowledge Plane, which is regarded as the central point of interaction between the intelligent logic implemented with GANA and the rest of the network systems, including the network elements, network management systems and any service management components that lie on top, comprising the whole service architecture. The Knowledge Plane contains the high-level policies and goals, and builds models that define how to achieve these goals network-wide. The Knowledge Plane works with the lower layers of the hierarchy of decision elements to implement the self-* behaviour in the network. This hierarchical positioning of DEs means that lower-layer DEs are seen as MEs by the upper-layer DEs. It needs to be noted that the GANA implementation does not require implementation of each of the hierarchical DEs layers; it is expected that layers will be added over time as the implementation matures and evolves.

When analysing the ETSI GANA architecture from the perspective of the ODA functional blocks, the main focus is on the Intelligence Management functional block, where the DEs and Knowledge Plane can be mapped. The MEs, on the other hand, such as network nodes, network management systems (NMSs) or architectural components that orchestrate end-to-end services or implement resource orchestration, are positioned in the Production functional block, as shown in Figure B.27. ETSI GANA is not related to any components that would belong in the Engagement Management or the Party Management functional domains.

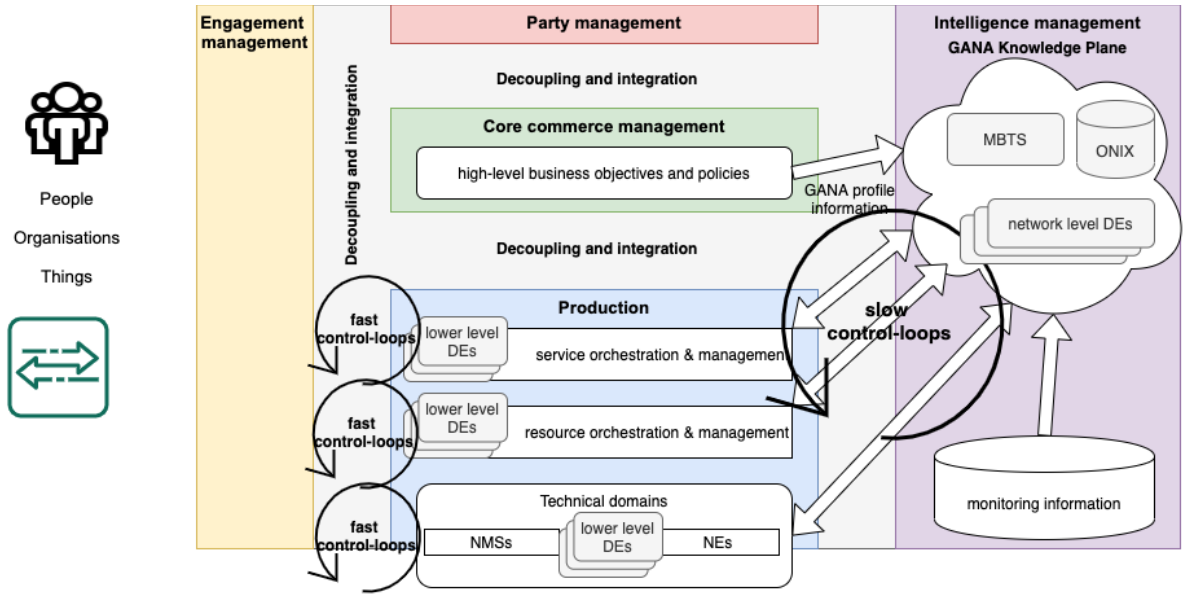

Figure B.27: ETSI GANA components mapped to the ODA functional blocks

Core Commerce Management

The business goals, product-level policies and organisation-wide objectives that represent the highest-level input for the GANA Knowledge Plane and DEs are fed from the Core Commerce Management functional blocks that store the information about products' specifications and expected behaviours, SLAs and other related policies. These can be considered as the global desired state that needs to be achieved with the self-* behaviour of the DEs.

Production

Similarly, the information stored in the Production functional blocks can be used to define some more fine-grained policies that are related to specific service implementations or resource types.

This block includes physical or virtual network nodes (or parts of network nodes) and their management hierarchy, including the network management systems, resource layer orchestration and management, service layer orchestration and management and overall end-to-end orchestration and management. Depending on the hierarchical layer of DEs, each of the components in these layers in the Production functional block can be seen as an ME and can be configured using autonomics by DEs. The lowest layer of MEs will be managed by the DEs using their exposed management plane interface or other objects for management based on a specific data model, management information bases (MIBs), for instance. Higher-layer MEs should be managed via their exposed REST APIs.

Having in mind the different timescales for control loops discussed above, it is evident that fast control loops will be implemented at the level of network nodes, while slow control loops will involve the orchestration layers on top.

Intelligence Management

In the Intelligence Management domain, the network-wide level of autonomic management is controlled by the GANA Knowledge Plane, which consists of several functional blocks: network-level

DEs, Overlay Network for Information eXchange (ONIX), and the Model-Based Translation Service (MBTS).

The DEs look at the network as a whole, using the network-wide information and desired state as their input to manage and balance the overall health of the network.

The ONIX represents a group of interconnected information systems that are tasked with gathering information and discovering DEs throughout the network. The ONIX's main purpose is information and resources gathering, thus this functional block does not implement AMC.

MBTS is the binding middleware between the language of the network elements that are being managed and the DEs that are doing the management. The purpose of this functional block is to translate from the network elements' data model, which is usually vendor specific, to a common, shared-information data model that is used by the DEs.

The GANA implementation has a network governance interface that network operators can use to create so-called GANA profiles, which are used as input to the Knowledge Plane. The GANA profile contains the network-wide goals, policies and high-level objectives. As mentioned above, this information is governed within the Core Commerce Management functional block, and it is expected to be fed into the GANA Knowledge Plane using automation tools. This interface is also used to feed the network configuration data and models into the Knowledge Plane, which can be done by providing information stored in the Production functional block.

DEs function at three further levels below the network-wide level: node level, function level and protocol level. On the node level, the DEs control the behaviour of a single network entity, including any orchestration and policies that the network entity is involved in, covering four main management aspects: security, fault, configuration and survivability. The function level DEs manage a group of functions of the network entity such as: routing, forwarding, QoS, mobility, monitoring and application. The lowest, protocol level DEs manage a single function; there is no collaboration between DEs on this level.

More information on the specification of the DEs' behaviour, the information model and reference points are available in the technical specification [AFITS18].

In the case of multi-network domains that implement multi-domain services, the ETSI GANA architecture supports a federated management approach that will implement multi-domain autonomics end-to-end. In this case, a special type of federated MBTS needs to be used so that it can translate the information from one domain to another. The federation approach defines that the DEs in one domain can only manage MEs in the same domain, whereas ONIXs and DEs from different domains can share the information required for the multi-domain use case.

B.2 Architecture Cross-Comparison

The selected architectures described and analysed using the ODA reference functional blocks have many commonalities in terms of design concepts, modules and implementation. All of the architectures have embraced the modular approach, using functional components to implement different aspects of the OAV architecture. Some of them strongly follow the loose coupling using REST APIs and microservices, where integration is achieved via orchestration and messaging between components, while some exhibit a less granular approach, with components that provide functionalities that may occasionally cross functional blocks. An overview comparison of the selected architectures is presented in Figure B.28.

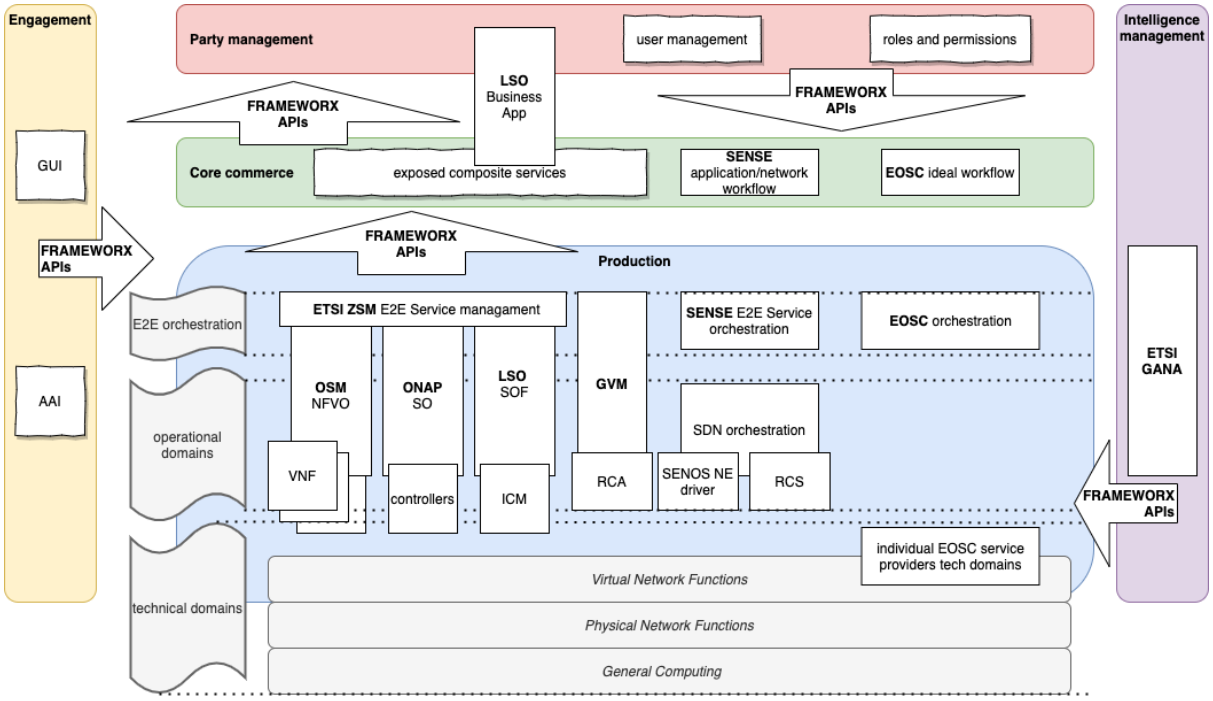

Figure B.28: Main focus of analysed architectures and their common components

Figure B.28 emphasises the main components of each of the architectures and highlights some of the architectural components that were commonly encountered in all examples (GUI, AAI, user management, roles and permissions, exposed composite services). As presented, all architectures have an implementation of at least a GUI for interaction with the end users, and in some cases additional APIs and CLIs are also available. In addition to the communication channel, there is also a commonly implemented AAI component which serves the purpose of identification, authentication and authorisation of valid users. The information about users is commonly stored in a separate component (a kind of CRM) that is used for user management and there is also a common notion of implementation of various roles and permissions, to serve as a role-based access control in the system. From the intelligence management point of view, in most of the architectures there is an implementation of a monitoring component that collects information about services and resources.

With the exception of the TM Forum Frameworx-based SPA, which provides a very detailed and highly granular definition of all functions and corresponding REST APIs for all identified components in each

of the functional blocks, most of the analysed architectures focus on the implementation of the Production functional block and the minimum set of components from the rest of the functional blocks that will support the system operations and management.

From the implementation point of view, ONAP is the most mature and complete of the analysed architectures, as it provides a very large number of implemented components for different aspects of an architecture, including the service design stage, the operational stage and the administrative management stage. A special-case example is the ETSI GANA architecture, which focuses solely on the implementation of the functionalities of Intelligence Management, as its purpose is to define a Generic Autonomic Network Architecture and thus provide a solution for the implementation of automated closed control loops.

When focusing on the Production functions, the analysed architectures' components are loosely grouped into verticals with their specific components. The focus of the EOSC architecture is to provide end-to-end orchestration and related functionalities for general-purpose computing services, connecting together the different computing infrastructures, including HPC, grid and cloud computing and data storage facilities. The implementation in each of these operational domains can be different, depending on the infrastructure provider, while EOSC tries to provide a layer that will enable interoperability between the infrastructures and make it possible for the end users to create the so-called EOSC ideal workflows consisting of services offered by different infrastructures. On the other hand, architectural solutions such as GVM and SENSE provide composite services that can be a mix of networking services and general-purpose computing, i.e. network testbeds or application workflows in SENSE. Both architectures can provide end-to-end orchestration of multiple operational domains, built on different approaches of resource abstraction and orchestration: combined for GVM and separate for SENSE. The main purpose of the rest of the analysed architectures is to manage network-oriented composite services.

All of the analysed architectures support a hybrid environment of physical and virtual networking functions, where the virtual network functions are implemented on top of a general-purpose computing infrastructure. The OSM, ONAP and LSO architectures are each capable of providing network service orchestration on the level of one operational domain and are also capable of orchestrating end-to-end services that span across multiple instances of operational domains. The ZSM end-to-end service management approach has been developed with a sole purpose of enabling the orchestration of multiple disparate operational domains, providing a common high-level orchestration approach for OSM, ONAP and LSO (and potentially other) architectures. The MEF LSO architecture focuses on the implementation of multi-domain services that span multiple independent network providers, defining the interaction between the providers, and also the interaction between the provider that offers the composite service and its customers, which is achieved via the business application component. On the other hand, the most popular architecture that is completely aligned with the ETSI NFV specification is the open source implementation of MANO, which has been used as a basis for, or can work with, many of the other analysed architectures and use cases (such as GVM, ZSM, GANA, etc.) and is therefore one of the most mature open source implementations available.

Table B.1 below summarises the main features of the architectures and emphasises their advantages and disadvantages.

Architecture	Key defining features	Pros	Cons
SPA based on TM Forum Frameworx	TM Forum Open APIs + Business Process Framework (eTOM) + Information Framework (SID)	+ Very high granularity + More than 50 APIs + Well-defined abstract information model + Open APIs specification available on GitHub	- Contribution to API definition and access to roadmap requires TM Forum Membership
MEF LSO	Multi-domain NaaS	+ Standardised lifecycles of end-to-end connectivity services across one or more network service domains + Detailed data model + Open source API spec	- Focus on specific service category (Carrier Ethernet services) - Small number of composite APIs
ETSI ZSM	E2E service management	+ E2E service provisioning in a transparent manner over different operational domains + Hierarchical structure with well-defined service points and interfaces + Full automation thanks to AI-based closed loops	- Not many implementations / deployments
ETSI OSM	Very widely used and deployed ETSI NFV-aligned MANO	+ Very active software release cycle (9 major releases in 4 years) + Full open source and commercial ecosystem available: Software Distributions, VNFs, VIMs and OSS/BSS + Commercial distribution, training and support available + Supports E2E and integrated orchestration + Supports technical and organisational multi-domains + Supports multiple VIMs (both VM and container-based) and WIMs (SDN-based) + Published interfaces and data models aligned with ETSI NFV which are routinely tested in ETSI NFV Plugtests with other ecosystem products	- Limited backwards compatibility (between first versions, now solved)

Architecture	Key defining features	Pros	Cons
Open Baton	Reference implementation, provides a centralised testbed view	+ Based on ETSI NFV MANO + Easy to install and extend + Open source	- Marketplace not available - Smaller community
GVM	Scalability, integration of any resource type	+ Deployment running as pilot service in DFN + Allows complex virtual objects that can be built iteratively using code + High flexibility for integration of resource types + Open source	- Multi-domain orchestration not fully available yet
SENSE	E2E multi-domain orchestrators Many-to-many relationship between orchestrators and resource managers	+ Flexible orchestration for individual application workflows	- Not deployed yet - Not fully implemented yet
ONAP	Design-time + Run-time + Management	+ Provides use cases (VoLTE, vCPE) + Includes a TOSCA- and YANG-supporting unified design framework + Open source	- Multi-domain orchestration not fully available yet - Some overhead as the result of two projects merging
EOSC	Cross-computational infrastructures interoperability	+ Federation of loosely coupled heterogeneous service providers	- Vision for full-scale deployment and development not fully clear yet - Requires synchronisation of a large number of projects and governance bodies
ETSI GANA	Autonomic network management	+ Can be combined with other architectures + Fast and slow control loops + Hierarchical approach + Multi-domain support	- Main focus is on Intelligence Management only - Needs a working, fully automated OAV solution to build upon - Time-consuming implementation - Customisation required

Table B.1: Key features, advantages and disadvantages of the analysed architectures

It is important to note, however, that the OAV core design principles discussed in Part A allow a solution that combines approaches, encompassing a number of the architectures presented in Figure B.28. By following the layered orchestration approach and using corresponding levels of service and resource abstraction, an architectural design can be created that will combine several of the architectures into one solution. Even an implementation of a recursive architectural reuse pattern is possible, as is presented in the next part of this document, Part C Mapping Use Cases and NREN Architectures to ODA.

Part C Mapping Use Cases and NREN Architectures to ODA

This part of the white paper reviews four use cases: two that are built using components from several of the investigated architectures and are implementing additional orchestration layers (5G and TALENT), as well as two examples of National Research and Education Network (NREN) architectures being mapped to the TM Forum Open Digital Architecture (ODA).

5G has been selected because of its specific features based on building recursive architectural solutions to implement different network domains. TALENT has been selected because it is the first solution that enables the unification of satellite and 5G terrestrial networks, in line with 3rd Generation Partnership Project (3GPP) and European Telecommunications Standards Institute (ETSI) standards. It has been implemented in the Satellite and Terrestrial Network for 5G (SaT5G) project for in-flight connectivity [SAT5G] and will be used in the RESPOND-A project for the first responders' connectivity [RESP].

The NREN mapping examples (CYNET and SURF) showcase how ODA can be used to analyse the current status of an NREN architecture and provide insight into future development.

C.1 Use-Case Mappings

C.1.1 5G

5G is a mobile technology that is addressing, by design, a broad range of requirements: very high data rates, very low latency, low energy consumption, high scalability, improved connectivity and reliability, and improved security [FGPPP18], [FGPPP20]. The 5G specifications focus on serving mobile operator needs in terms of extreme mobile broadband services and include several features to support vertical industries in terms of enablers for an industrial Internet of Things (IoT) and Ultra Reliable Low Latency Communications (URLLC) [FGPPP20]. This section presents orchestration, automation and virtualisation (OAV) elements of the 5G architecture. Within the 5G system, end-to-end (E2E) network slicing, service-based architecture, Software-Defined Networking (SDN) and Network Functions Virtualisation (NFV) are seen as the fundamental pillars of the 5G architecture. To this end, the 5G

Infrastructure Public Private Partnership (5G PPP) specified and developed the main elements of the 5G architecture, shown in Figure C.1.

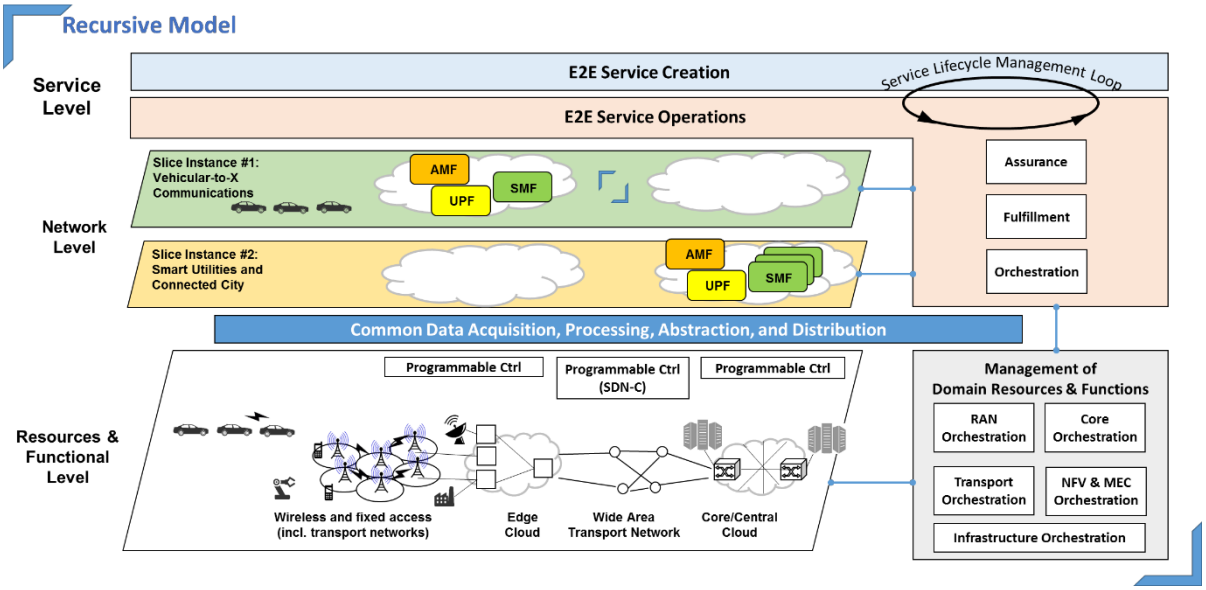

Figure C.1: 5G overall system architecture [FGPPP16]

Network slicing in 5G requires the ability to execute multiple logical mobile network instances on a collective infrastructure. In 5G, SLAs should be in place for each customer. E2E frameworks for Service Creation and Service Operations have to exhibit a significantly increased level of automation for the lifecycle management of Network Slice Instances (NSIs). On the Service Level (see Figure C.1), lifecycle management automation has to be realised by closed-loop Service Assurance, Service Fulfilment and Service Orchestration functions covering all lifecycle phases (i.e. preparation, instantiation, configuration and activation, run-time and decommissioning). The enabling technologies that help in the overall orchestration are virtualisation of network functions, as well as software-defined, programmable network functions and the new innovative infrastructure resources that are available. E2E Service Operations functions cooperate with functions for Management of Domain Resources and Functions such as Network Function Virtualisation (NFV) and Multi-access Edge Computing (MEC). Also, for orchestration, closed-loop procedures for resource fulfilment, resource assurance and network intelligence comprise the building blocks within each management domain. SDN controllers can be programmed to efficiently execute policies and rules on the Resources and Functional Level. 5G supports a common platform, where data can be accessed by system entities from all levels. It uses scalable data exposure governance and access control mechanisms to provide services for data acquisition, processing, abstraction and distribution of data related to subscribers, to the network and underlying resources, to network slice and service instances and applications. A range of stakeholder roles are defined in the 5G ecosystem, as given in [FGPPP16], namely:

- **Service Customer (SC)**: a person or organisation that uses Service Provider (SP) offerings.
- **Service Provider (SP)**: an organisation that offers 5G telecom services, digital services (i.e. enhanced mobile broadband, IoT) or provides network slices as a service.

- **Network Operator (NOP)**: orchestrates resources from multiple Virtualisation Infrastructure Service Providers (VISPs). The NOP utilises aggregated virtualised infrastructure services in order to design, build and operate network services that are offered to SPs.
- **Virtualisation Infrastructure Service Provider (VISP)**: provides virtualised infrastructure services comprised of networking and computing resources.
- **Data Centre Service Provider (DCSP)**: provides data centre services. A DCSP differs from a VISP by offering "raw" resources services. A VISP offers access to a diversity of resources by accumulating different technology domains and making them accessible in a single API.

In order to serve all aspects of network slicing, the 5G architecture is divided into different layers (functional blocks) [FGPPP14], [FGPPP18]:

- The **Service layer** comprises Business Support Systems (BSSs) and business-level policy and decision functions, as well as applications and services combined as products offered to a tenant (end-to-end orchestration system included).
- The **Management and Orchestration (MANO) layer** includes ETSI NFV MANO functions, such as the Virtual Infrastructure Manager (VIM), the Virtual Network Function Manager (VNFM) and the Network Function Virtualisation Orchestrator (NFVO). The ETSI NFV MANO has an Inter-slice Broker interacting with the Service Management function for handling cross-slice resource allocation actions. Furthermore, the MANO layer allows domain-specific functions for application management (e.g. Element Managers (EMs) and Network Management (NM) functions, including Network (Sub-)Slice Management Functions (N(S)SMFs)). These functions implement ETSI NFV MANO interfaces on the VNFM and the NFVO. Service Management is a function between the Service layer and the Inter-slice Broker (extending to all network slices) that transforms consumer-facing service descriptions into resource-facing service descriptions and vice versa. The Domain Manager, the Inter-slice Broker and NFVO together comprise the Software-Defined Mobile Network Orchestrator (SDM-O), which is accountable for the end-to-end management of network services. SDM-O can set up slices and merge them appropriately at a defined multiplexing point using the network slice templates.
- The **Control layer** puts together the two main controllers of 5G, i.e. Software-Defined Mobile Network Coordinator (SDM-X) and Software-Defined Mobile Network Controller (SDM-C), as well as other control applications. The SDM-C and SDM-X can be executed as Virtual Network Functions (VNFs) or Physical Network Functions (PNFs) themselves and can handle dedicated and shared NFs respectively according to SDN principles, by translating decisions of the control applications into commands to VNFs and PNFs.
- The **Multi-Domain Network Operating System Facilities layer** consists of adaptors and network abstractions used "above" the networks' and clouds' heterogeneous fabrics. It is responsible for allocation of the required virtual network resources and the stability of the network state in order to ensure network reliability in a multi-domain environment.
- The **Data layer** comprises the VNFs and PNFs and handles the user data traffic.

Figure C.2 provides a summary overview of how the 5G architectural functional blocks described above can be mapped to ODA functional blocks.

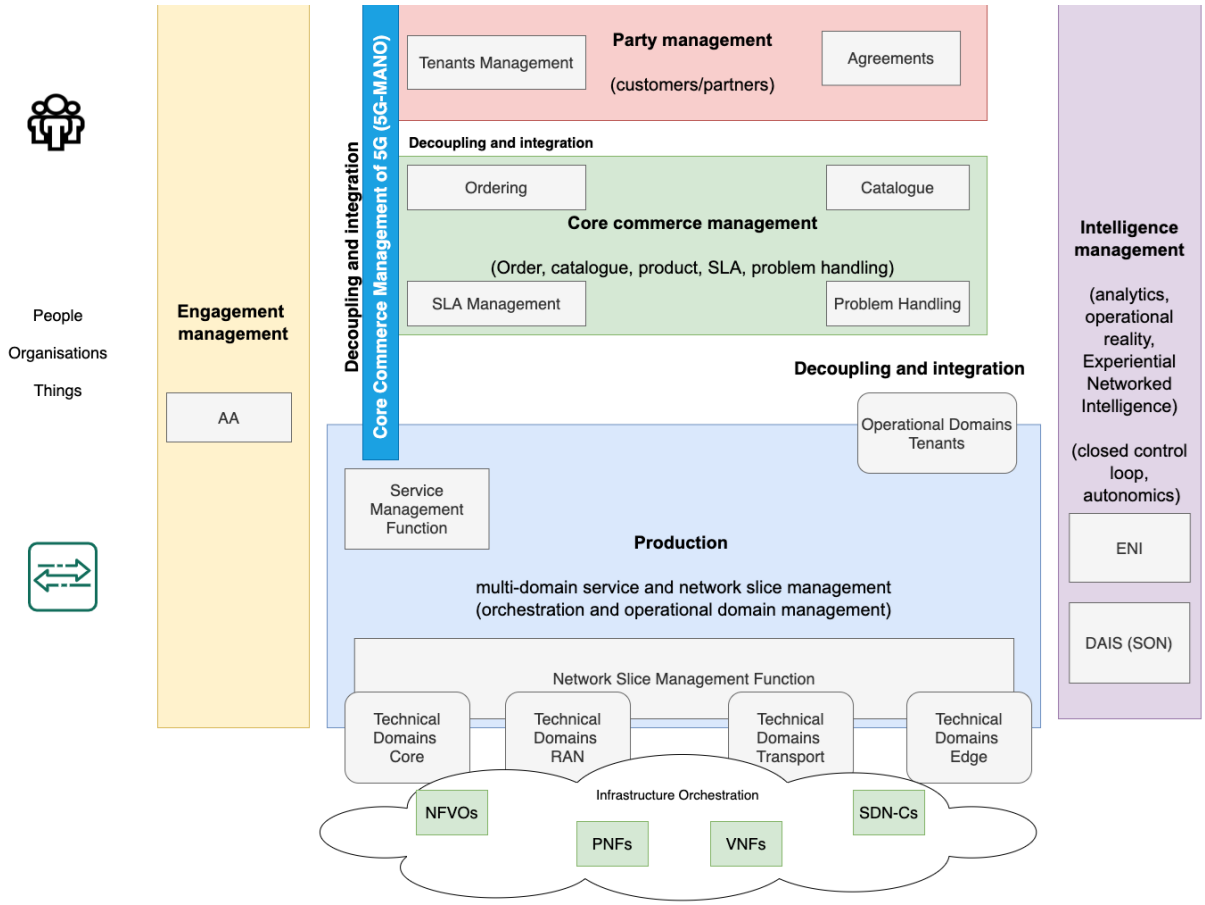

Figure C.2: 5G architecture mapped to the ODA functional blocks

Engagement Management

The Engagement Management domain includes the Authentication and Authorisation (AA) API [FGPPP14]. AA is an intermediate component that works closely with the API to identify end users through a credential-based authentication process. After authentication is completed, the AA component evaluates and authorises the claims of the end user within the system. In AA, each user can have access to different orchestrator components, in different administrative domains, through the authorisation module (e.g. Multi MANO Manager).

Party Management

Party Management is executed through business relations [FGPPP16], [FGPPP18]. The business relations among Managed Service Providers (MSPs), Infrastructure Providers (InPs) (providers of integrated photonic circuit technologies) and tenants have an impact on the architecture. Therefore, this module has a tight interaction with the Multi MANO Manager (shown as 5G-MANO in Figure C.2) and Domain Manager in order to store, retrieve and check, as well as organise, the user information. It is responsible for keeping all necessary information about users, organisations and interaction procedures of the supported NFVO, VIM and Radio Access Network (RAN), including IP address, port number, project, tenant and connection setting. By keeping this information, users can manage and handle operations and services across the system and over underlying solutions. Each tenant in 5G signs a service-level agreement (SLA) with the Mobile Network Operator (MNO), in order for the provider to manage its available resources and meet the individual tenant requirements and any

excess requests that the tenant requires. The management of the resources is handled in a recursive and hierarchical manner through an API offered by the Multi-Tenancy Application (MTA), in conjunction with the defined operations and policies on top of the virtual infrastructure. Therefore, each tenant operates its virtual infrastructure the same way the MNO operates on the physical infrastructure, by allocating and reselling part of the resources to other Mobile Virtual Network Operators (MVNOs) in a transparent way to the MNO. In addition, the API exposed to the end users provides the capability, over the cloud and edge computational resources, to deploy end-to-end 5G services.

Core Commerce Management

The Core Commerce Management block of 5G, through the 5G-MANO, illustrates how automatically the architecture can provision the required end-to-end service (composed of cloud/edge and RAN). More specifically, 5G is a layered recursive architecture [FGPPP16], [FGPPP18], [FGPPP20]. In the lower layer, the owner of the physical resources (MNO) instantiates the aforementioned resources in its MANO. On top of the physical infrastructure, diverse tenants request allocation of virtual infrastructures composed of a network subset with virtual nodes and links (i.e. a slice) through a Multi-Tenancy Application (MTA), which orchestrates the allocation of the free resources. The MTA requests from the Virtual Infrastructure Manager (VIM) the creation of a virtual topology according to the tenant's demand. In order to meet all kinds of tenant demands, 5G offers different services involving different Service Provider (SP) roles: Communication Service Provider (CSP) offering traditional telecom services; Digital Service Provider (DSP) offering digital services such as enhanced mobile broadband and Internet of Things (IoT) to various vertical industries; and Network Slice as a Service (NSaaS) Provider offering a network slice along with the services that it may support and configure. Therefore, 5G offers to SPs the availability to design, build and operate services using aggregated network services.

5G also provides over-the-top (OTT) services. An example of such services between Tenant#1 and Tenant#2 is where the Tenant#2 MVNO infrastructure operates over the virtual network that is provided by Tenant#1 MVNO, which operates on top of the physical infrastructure of the MNO. In this example, the MTA is required to provide the tenant identification, while the mapping of the virtual to physical resource is handled by the VIM through the NFVO, in different administrative and technical domains. The architecture demonstrating this example is presented in Figure C.1. In addition, the 5G architecture supports the deployment of the network services of the tenants that have OTT services on top of all other tenant services, by extending the functions of the NFVOs to support tenant separation and identify the mapping of a tenant to a network service comprising a set of VNFs connected in a forwarding graph, in a similar way to Open Source MANO and Open Baton. This architecture follows the recursion principles of the ONF architecture.

The technical dependencies among stakeholders are diverse. For example, MSPs rely on the level of resource availability and utilisation information provided by the InP(s), exposed through APIs from the VIM, to control and administer the network functions and services. In addition, both the tenant and the MSP depend on the SLA in place between them, for example in order for the MSP to give the tenant partial access to NFV MANO layer functionality with the use of APIs for specific entities (e.g. SDM-O, SDM-C, and Service Management) or through dedicated instances of these entities, controlled by the tenant. Further, the tenant itself can provide network function software, offered through the VNFM and Network Management entities. The MANO and control framework should handle requests from VNFs and VNFMs belonging to different entities or organisations, and should coordinate the requests with the SDM-O/SDM-C. Consequently, 5G can be seen as a modular, multi-tenant, multi-

network architecture, where different stakeholders/tenants can interact. The implementation of legally binding agreements is therefore essential. Keeping track of SLA terms between parties is a challenging and demanding task; the MANO and control framework are responsible for storing and monitoring all SLA agreements between involved parties.

Production

5G focuses on the management of resources in cloud and radio infrastructure by following the multi-tier orchestration concept [FGPPP16], [FGPPP18]. 5G uses Domain Managers and coordinates their actions. 5G has two major operational stages: creation of the Network Slice Instance (NSI)/bootstrapping and run-time. The first phase consists of setting up the security credentials in the 5G system to be ready for the correct operation over an infrastructure. The MANO Selector Plugin and Domain Component Plugin are loaded with inputs based on the Tenant SLA for the supported MANO and VIM. These inputs will be used afterwards at the run-time phase. The run-time phase is responsible for execution of operational commands coming from diverse stakeholders. In the run-time phase, the network slicing services take control. Also, there are two main network slicing services that enable different degrees of explicit control and give different levels of automation of the mobile network slices management:

- **Virtual Infrastructure (VI) service**: offers control and operation to the different tenants in line with the Infrastructure as a Service (IaaS) model (i.e. creation of a Network Slice Instance).
- **Tenant-owned Network Service (NS):** the same service as defined by the ETSI NFV architecture (i.e. creation of a Service Instance).

In the VI service, the deployment of a mobile network is executed with the allocation and deallocation of VIs. More specifically, a VI is a logical construct composed of virtual links and nodes, which acts as a physical infrastructure. The logical entities in a VI encompass a set of compute and storage resources interconnected by a virtual and logical network. The VIs can be controlled and operated by the tenant through different SDN control models, enabling different degrees of internal control. This service handles dynamic allocation, operation and deallocation of a VI. In order to provision the VI, a direct hardware element support, or its emulation via software for multiplexing over the mutual infrastructure, is needed.

Network Services (NSs) are instantiated on top of the shared infrastructure, and comprise a set of interrelated Virtual Network Functions (VNFs). The virtual services exploit sharing the computing, storage and network resources of a common physical infrastructure. Thus, on a tenant's request for a service, there is usually an indication of the kind of VNFs to be used, through the NS Descriptor; of the capabilities and scope, through one or more VNF Descriptors; and how they must be interconnected, through a VNF-FG Descriptor. To ensure the use of common NS templates, description of the information elements is currently under standardisation process in the ETSI NFV ISG, in OASIS TOSCA standards and in IETF.

Additionally, 5G provides high-level mechanisms for multi-domain configuration network slicing in order to support a service that is established throughout a collaboration of the domain-specific orchestrators, based on policies and agreements that are applicable over the participating administrative domains.

Intelligence Management

5G-MANO supports the dynamic deployment and service-aware adaptation of the applications by using a set of optimisation schemes and intelligent algorithms that select the needed infrastructure resources from different domains. The following procedures are relevant:

- **Network Slice Instance (NSI) deployment**: auto-deployment using OpenStack with intelligent algorithms.
- **Network Slice Instance monitoring**: each system component will register itself to an internal discovery service, which controls the activity status of each component.

In addition, taking advantage of cutting-edge technologies in the area of artificial intelligence, a Self-Organising Network (SON) decision-making framework is proposed to provide a possible method to realise intelligent management for the upcoming 5G networks. There are cutting-edge technologies that utilise BDIx agents in a Distributed Artificial Intelligence (DAI) way in order to implement distributed SON in device-to-device (D2D) communications at 5G [DAIS20]. The Experiential Networked Intelligence Industry Specification Group (ENI ISG) is defining a Cognitive Network Management architecture, using AI techniques and context-aware policies to adjust accessible services based on changes in user requirements, environmental conditions and business goals. It consequently fully benefits the 5G networks with automated service provision, operation and assurance, as well as optimised slice management and resource orchestration. ENI has also launched proofs of concept (PoCs) showing how AI techniques can be used to help network operation including 5G [ENI20].

C.1.2 TALENT

Terrestrial Satellite Resource Coordinator (TALENT) is a coordination tool which provides end-to-end services over satellite domain, radio system, cloud and Mobile Edge Computing (MEC) resources [TAL19]. It provides a single and easy-to-use point of interaction for all stakeholders involved in the ecosystem, such as terrestrial and satellite operators as well as different 5G verticals. TALENT is completely in line with 3GPP and ETSI definitions, extending them towards satellite systems.

The original idea of TALENT is based on the definition of hierarchical and distributed orchestration [HDO17], where an overarching orchestrator is able to manage and coordinate several independent domains (satellite, radio and cloud). It is assumed that a domain manager (DM) exists in each domain that can work with the resources of this domain. In this sense TALENT becomes a light, scalable and efficient solution, agnostic to elements of domains coming from various vendors.

Having these objectives in mind, and based on the frameworks suggested by ETSI MANO [ETSIMANOA] and 3GPP SA5 [TGPPR14], this work proposes an extension towards satellite integration at the management and orchestration level, to build a multi-tier orchestration stack over a heterogeneous environment.

As shown in Figure C.3, the first release of TALENT supports satellite and cloud/edge domains, interworking with each other to deliver end-to-end services. A new release will follow [TGPPR14] in order to introduce the radio domain to TALENT.

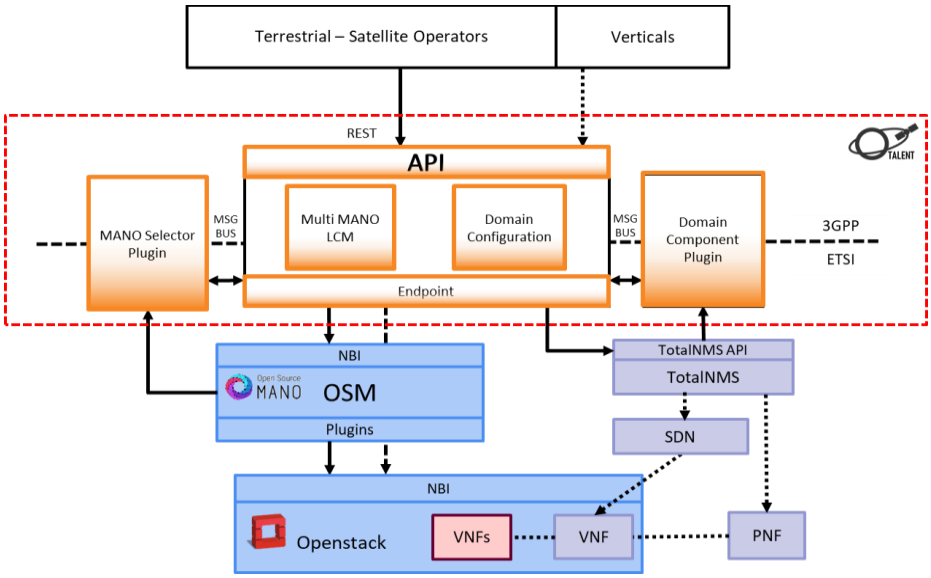

Figure C.3: TALENT architecture

The internal components of TALENT, shown in Figure C.3, are described in more detail below:

- **Northbound REST API**. This is the main entry point for different actors such as operators and verticals. It also provides an abstraction layer, which exposes a well-defined set of functions serving the different needs of operators and verticals (e.g. service instantiation, check service status, quality of service (QoS) monitoring). The TALENT API is a REST-based API, which feeds the TALENT graphical user interface (GUI).

- **MANO Selector Plugin**. This module is a reference point where all the required information, such as IP address, data model format, vendor and interaction procedures, of the supported NFVOs (OSM, ONAP, etc.) and VIMs (OpenStack, OpenVIM, etc.) are kept. The MANO Selector Plugin is responsible for registering all supported NFVOs and VIMs at the bootstrapping phase. Release 1 supports OSM release 4 and 5, and OpenStack Queen. In the next releases, other NFVO (e.g. ONAP) and VIM (e.g. OpenVIM) solutions will be added to TALENT.

- **Multi-MANO Lifecycle Manager**. This module is responsible for supporting NFVO and VIM clients, including their actual service and lifecycle management in the run-time phase. It uses a template file to produce requests with required attributes to the cloud/edge domains that are registered to the system. To manage all received requests, the Multi-MANO Lifecycle Manager synchronises with the MANO Selector Plugin to load the required dependencies and interact with underlying NFVO and VIM solutions.

- **Domain Component Plugin**. This module is a reference point to keep all the required information, such as IP address, data model format and configuration settings, of the supported satellite components. The Domain Component Plugin is responsible for registering all supported satellite components at the bootstrapping phase. TALENT will be loaded with required dependencies and libraries to establish communication with underlying satellite solutions. Release 1 supports TotalNMS of Gilat [TNMS20]. Later, more vendors will be added to TALENT.

- **Domain Configuration Module**. This module is responsible for supporting satellite clients, including configuration and lifecycle management of supported solutions in the run-time phase. Similarly to the Multi-MANO Lifecycle Manager, it uses a template file to produce requests with required attributes to the satellite domains that are registered to the system. To manage received requests, the Domain Configuration Module synchronises with the Domain Component Plugin in order to load the required dependencies and interact with underlying satellite solutions.

Figure C.4 presents a mapping of the TALENT architecture to the TM Forum ODA architecture.

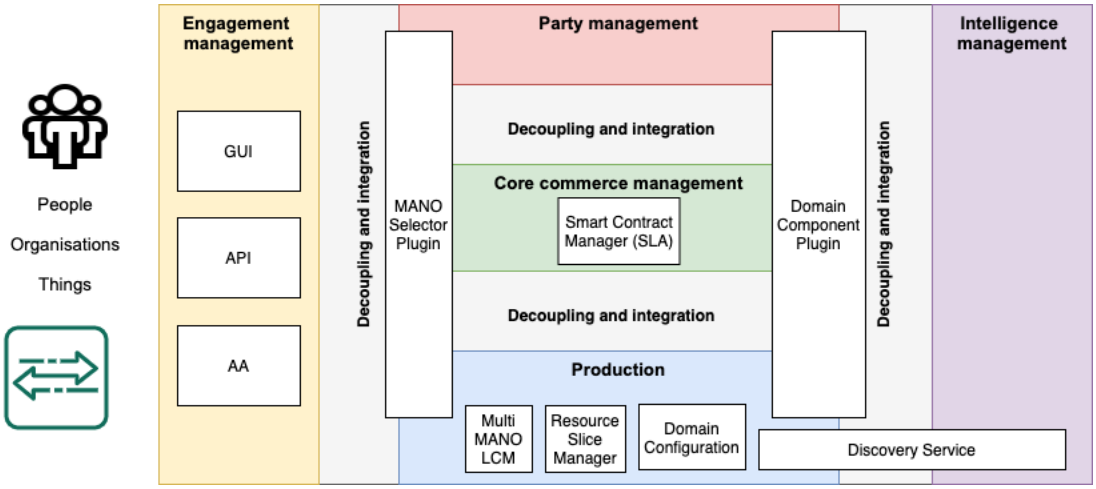

Figure C.4: TALENT architecture mapped to the ODA functional blocks

Engagement Management

In TALENT, the Authentication and Authorisation (AA) API and GUI are part of Engagement Management. AA is an intermediate component that works closely with the API and GUI to identify end users through a credential-based authentication process, and evaluates the claims of the end user within the system that occur after the authentication. The API and GUI exposed to the end users provide the capability to deploy end-to-end 5G services upon the cloud and edge computational resources.

The Engagement Management module identifies and authenticates the user when interactions between different orchestrator components are performed, especially when these components reside in different administrative domains. It also checks the authorisation of a component (for a given user/role) to trigger certain tasks implemented by a different component (e.g. Multi-MANO Lifecycle Manager) by calling the interfaces of the latter. The engagement module tends, as far as possible, to remain user friendly and interoperable with the local NMS users and the TALENT GUI.

Party Management

Together with Production, the Party Management module has a tight interaction with the Multi-MANO Lifecycle Manager and Domain Configuration in order to store, retrieve, check and organise the vendor information. It is responsible for keeping all required information about users, organisations and interaction procedures of the supported NFVO, VIM and RAN, and satellite proprietary data, including IP address, port number, project, tenant, connection setting and other

specific information. By keeping this information, users can manage and handle operations and services across the system and over underlying solutions. In addition, it holds different views to enable multi-tenancy by allowing different roles and capabilities for each user and tenant.

Core Commerce Management

Automatic provisioning of the end-to-end TALENT services composed of cloud/edge, RAN and satellite is done in the Core Commerce Management domain. With the use of content multicasting, TALENT demonstrates efficient delivery of refreshable and live content with the help of 5G technology integrated with satellite systems.

TALENT represents a modular, multi-player ecosystem, where different stakeholders can interact. For this purpose, it is important to keep track of any SLA terms between parties and make sure that the contractual agreements are met. The Core Commerce Management domain is responsible for keeping all SLA agreements between involved parties with the help of technologies such as Blockchain, and, based on the telemetry information extracted from the running services, it applies an automatic penalty/bonus mechanism. SLAs have to be loaded to this module during the bootstrapping phase. During this phase, the loaded information will also include a payment account where the automatic penalty/bonus mechanism will be applied.

Production

TALENT is a light solution focused on the coordination of resources in three domains: cloud, radio and satellite. Following the multi-tier orchestration concept, instead of handling everything by itself, TALENT uses domain managers and coordinates their actions. From one side, vendors can keep their specific management solution proprietary (good for keeping the business) and from the other side, they can expose APIs towards TALENT, and thus contribute to the end-to-end service offering. It is a top-layer solution, modular and extensible. As the solution has to be completely agnostic with regard to the underlying layers, the architecture has to offer a high level of dynamism when it comes to supporting different frameworks, tools and proprietary solutions.

TALENT has two main operational phases: bootstrapping and run-time. Bootstrapping consists of setting up the TALENT system to be ready for the correct operation over an infrastructure. It is a one-time process (every time the system starts afresh). The MANO Selector Plugin and Domain Component Plugin are loaded with inputs for the supported MANO, VIM and satellite solutions. These inputs will be used later on in the run-time phase. The run-time phase is responsible for execution of operational commands coming from different stakeholders. In principle, TALENT supports two categories of operational commands:

- **Network Service-related commands**. These are commands for lifecycle management of end-to-end connectivity and computational resources. TALENT eases the provisioning/termination of end-to-end services over satellite and terrestrial resources.
- **5G application-related commands**. Over the provisioned network services, TALENT is able to run and manage different 5G applications.

Intelligence Management

TALENT supports auto-deployment practices, which means the user does not need to go through a complex process to install TALENT. Please note that the auto-deployment does not include any closed

control loop behaviour and is partially located in the Intelligence Management and Production modules. The following procedures are relevant:

- **System deployment**. TALENT supports auto-deployment using containers (Docker-based). An ordinary container management tool for running multi-container applications can easily start up the system and its components.
- **System check**. Each system component will register itself to an internal discovery service (Consul.io), which controls the activity status of each component and TALENT overall. This process runs when TALENT is freshly installed in a new system. Upon deployment of TALENT components from the Docker container, all of them register themselves automatically in the discovery service. The discovery service creates a table with the name of the component and its status (READY, UNKNOWN). If the status of all components changes to READY and they are correctly linked together, then TALENT endpoints become available (API and GUI).

Consul.io is responsible for providing consistent and available information about the modules, including service and node discovery mechanisms, a tagging system, health checks, to name a few possibilities. Leveraging Consul.io within the system allows a sophisticated level of awareness to be built into the applications and services. It enables a module to know about the status (querying by their names) of running and deployed modules in the system. Once all the modules are registered in Consul.io, they will be ready to accept inputs and start their processes to provide services to the underlying MANO and satellite domains.

The system check functionality enables the gathering of health data about the overall system that can help administrators react to events accordingly. The implementation supports future upgrades using closed control loops that will provide self-healing and auto-scalability functionalities.

C.2 NREN Mappings

The NREN community provides another source of interesting use cases when it comes to a reference architecture such as ODA. NRENs in GÉANT are very diverse, ranging from serving small islands to large countries with extensive networks, and thus operate different architectures based on varying levels of OAV implementations. A good reference architecture should work for all of the NRENs, allowing them to map their architectures to the blueprint as a baseline for possible collaboration by identifying the same functional blocks and possible open APIs. Currently the architectures of two NRENs have been mapped to the ODA reference architecture: CYNET and SURF, presented in the following sections. Any NREN interested in mapping their architecture to ODA should please send a request to oav@lists.geant.org.

C.2.1 CYNET

CYNET is Cyprus's National Research and Education Network. It provides a network infrastructure for the Cypriot research and education community and connects research and education institutions. CYNET-II, the national backbone of CYNET, is connected to the European backbone GÉANT, which is part of the worldwide community of research and education networks. Because Cyprus is a small island, CYNET serves only a few universities (around 17), which has so far allowed its workflows to be mainly non-automated processes. However, although CYNET is currently only at the beginning of its journey into orchestration, automation and virtualisation (OAV), going forward the organisation would like to do more to benefit from adopting OAV principles, and to make sure it is compatible with future OAV multi-domain processes and workflows throughout Europe.

A mapping of CYNET's OAV architecture to ODA is shown in Figure C.5 below.

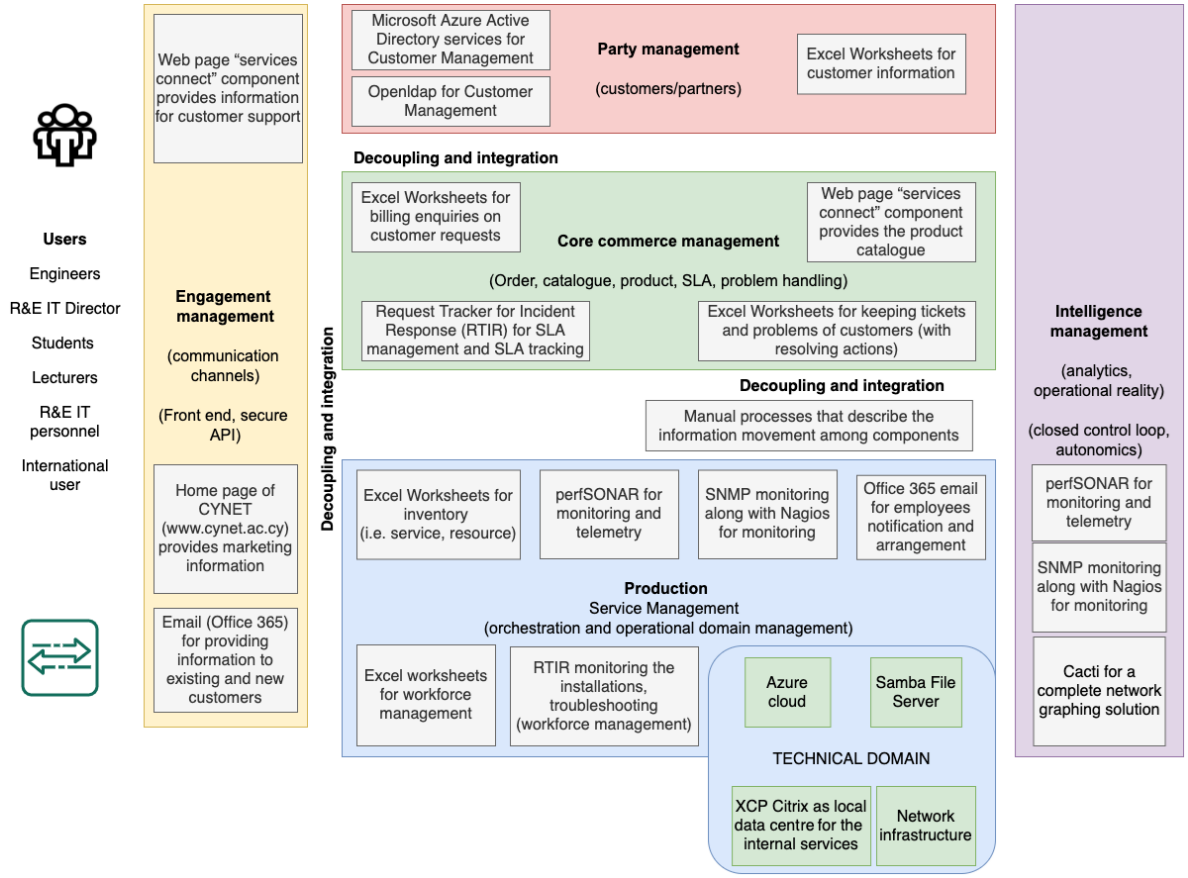

Figure C.5: CYNET OAV architecture mapped to the ODA functional blocks

When compared to the ODA core concepts and design principles, it is evident that the CYNET architecture has considered and adopted some similar ideas and approaches, e.g. using a service management process based on Request Tracker for Incident Response (RTIR) that is not only responsible for the implementation and lifecycle of the service, but is also used to put monitoring of the service and SLA tracking into place. The RTIR ticket system is part of Core Commerce Management

in relation to ODA core concepts. To see how CYNET's architecture can be mapped to all the functional blocks of the ODA architecture, please go to [CY2020].

C.2.2 SURF

SURF, the Dutch NREN, is one of the NRENs in the GÉANT community that has been focusing on improving and advancing its network architecture and management by introducing automation and orchestration in all workflows relating to network services, while at the same time working on reconstructing the network at the hardware level.

When placed in the context of the TM Forum ODA functional representation, the SURF OAV architecture can be represented as shown in Figure C.6. The grey boxes in the diagram represent SURF architecture components, and their placement within the ODA functional blocks is defined based on their main functionalities.

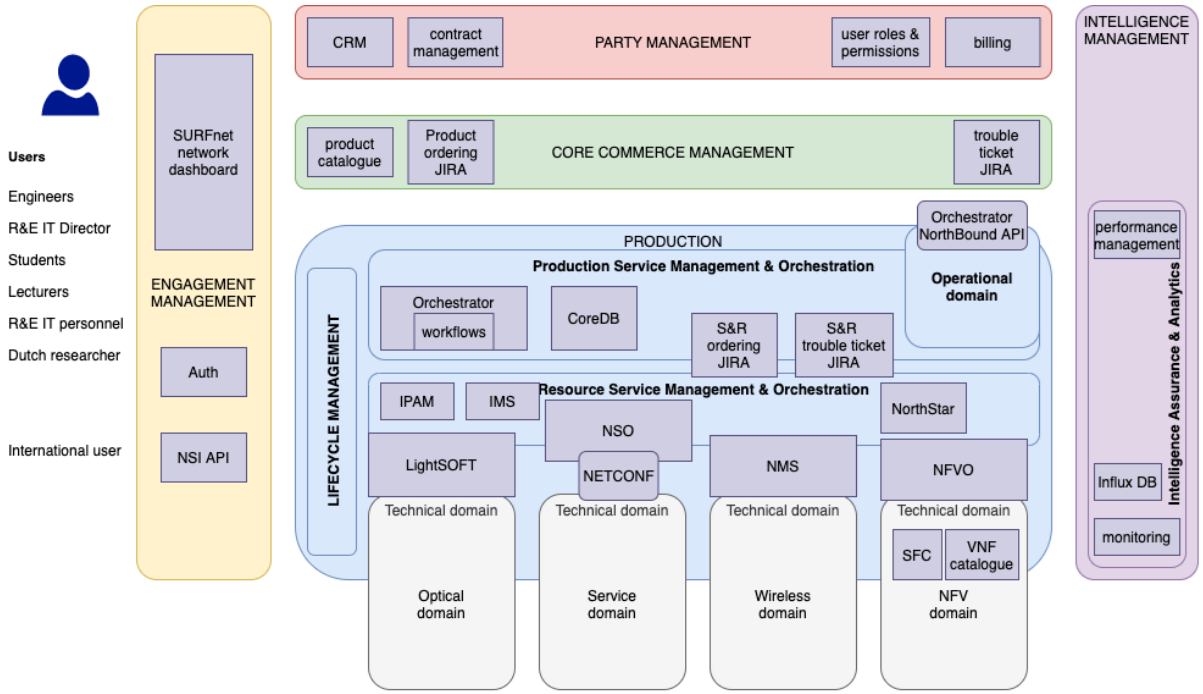

Figure C.6: SURF OAV architecture mapped to the ODA functional blocks

For more information on how SURF's architecture maps to ODA, please see [SUR2020].

Conclusions

Recent technological advances that focus on exploiting the possibilities for orchestration, automation and virtualisation (OAV) have induced new approaches in the design of digital services architectures. A flexible and agile environment that can easily be augmented and connected with others is recognised by digital service providers as the cornerstone for building solutions and platforms that will respond to the changing demands of their customers. These changes are not only felt in the commercial, competitive digital market, but are also reflected in the R&E environment. The GÉANT community and the NRENs are facing similar changes on the demand side and, therefore, need to evolve their architectures in order to be able to adapt and remain on the cutting edge of the service delivery landscape.

This white paper aims to help the community understand the changes and the needs for the implementation of novel approaches that will embody the principles of orchestration, automation and virtualisation. It presents the requirements for the new generation of OAV architectures and the design principles and key considerations that need to be put in place so that NRENs can transform into agile and flexible digital service providers. The GÉANT community has already been working on multiple aspects of OAV architectures, and the NRENs are at different stages on their paths towards the implementation of OAV within their own domains, some very far advanced, but many still at the beginning. The NREN survey run by the WP6 OAV team [GSUR19] has shown that having a common terminology and reference architecture will improve the potential for achieving a common understanding between NRENs and active project working groups and for facilitating collaboration and future interoperability.

Based on the analysis of the digital landscape, a list of key OAV architectural functionalities has been provided that highlights the main considerations and aspects needed in this process of transformation. In essence, a new digital OAV architecture needs to be modular, comprising a number of loosely coupled components that share a common information model; the service lifecycle should be processed in a standardised abstracted manner supporting a multi-vendor approach; automation should be employed as much as possible; and the solution needs to be built with the possibility for flexible integration with other domains. The TM Forum Open Digital Architecture (ODA) has been chosen as a reference architecture for the reasons given in Part A, but in particular for its modular design concepts and functional block grouping, common terminology and principles.

A selection of OAV architectures from standardisation bodies, organisations and research projects has been mapped in this document against the ODA blueprint architecture. Individual components with specific capabilities are categorised into ODA domains, thus enabling the reader, knowledgeable in at least one of them, to more easily understand other architectures and – more importantly – to understand the similarities and differences among them.

Additional use-case examples present the possibility to combine multiple architectures into one solution, following a layered architectural approach where specific aspects or operational domains can be implemented using different architectural solutions.

Thus, the purpose of this white paper is not to propose or choose one of the presented architectures that should be followed by the community. On the contrary, the main idea is to show the commonalities and to identify the generality of all of the analysed approaches, which allows each NREN to choose its own path towards OAV, including the architecture design, allowing freedom with the specifics of internal implementation, but at the same time following a core set of design principles that will enable interoperability and flexibility in building multi-domain solutions.

The analysis of the selected architectures underlines the importance of the modular approach combined with abstraction layers. All architectural solutions are built on the main OAV requirements, consequently having many common components and approaches in their implementation. The concept of decoupling of functionalities and integration of functional components using REST APIs and orchestration capabilities is another common element across the architectures. This approach not only enforces the flexibility of the solution, but also provides a way to build a common abstraction layer that will enable the support of a multi-domain environment built on the premises of a common information model and common set of exposed APIs. In addition, the use of catalogues and service discovery mechanisms allows dynamic and fast implementation of new services, either elementary or composites of multiple supporting services. The hierarchical process of building services using abstract service elements and resources also provides a flexible way to build on top of disparate technological domains that can be composed of multi-vendor equipment. A common way to address this problem is to use a high-level orchestration for end-to-end service orchestration with a lower-level orchestration process for implementing the service within each separate technological domain. The use of APIs enables this whole process to be fully automated and, in addition, even controlled by the end user, if this is a feature that the provider would like to expose to customers. In other words, the abstract service and resource composition allows fine granular control over what is visible to user groups and domains, while enabling a standardised way of handling requests and managing assurance and fulfilment processes across different technologies.

Therefore, the infrastructure on top of which these OAV architectures run can be a hybrid, built as a combination of physical network elements and virtual network elements embodying Network Function Virtualisation (NFV) concepts. Moreover, the infrastructure can also provide common computational virtual resources, thus enabling the definition of not only pure network services, but general digital services that are built as a combination of networking and computational resources. Using the same approach of service and resource abstraction and APIs for both network and computational resources enables this transparent orchestration of composite digital services, which is one of the major characteristics of the new digital architectures.

Another important aspect that needs to be addressed when building an OAV architecture is the implementation of intelligence management. While at this moment the approaches to implementation of autonomic networks and automated closed control loops are still in their early days, efforts are being made towards standardising the management and use of the knowledge gathered via monitoring and logging. This means that new OAV architectural solutions should incorporate functionalities that enable storing and management of big data and AI-based functionalities that will serve as control-feedback loops for automated self-healing networks. The (suitably controlled) sharing

of such data between domains, for example between NRENs, via appropriate APIs, is also likely to add value.

Using the TM Forum ODA as a reference point in terms of terminology, the definition of components and their functionalities and functional groupings can alleviate this problem and enable NRENs to find common ground when discussing different implementations of the same modules. Following the main ODA core design principles – built on the premise of a building-blocks-based architecture that decouples functionalities into components accessible via well-defined APIs and integrates these functionalities using orchestration approaches – will help NRENs build their customised OAV solution that will be tailored to their own needs and purposes. In addition, the implementation of service and resource abstraction and hierarchical composition will enable NRENs to have full control over the information, services and resources which are exposed to their customers or partner domains. At the same time, the possibility to expose parts of their catalogues and inventories will enable them to integrate into a multi-domain ecosystem for which a small set of well-defined APIs and corresponding information models need to be agreed among the partners.

The approach in this white paper, where ODA is used as a reference point to analyse different architectures, is an example of how common terminology and understanding can be built among the different NREN architectures and approaches. Furthermore, one of the future goals of the OAV architecture focus group within WP6 T2 is to continue its efforts by applying ODA to analyse additional current NREN architectures.

References

[AFITS18] ETSI TS 103 195-2 Technical Specification. Autonomic network engineering for the self-managing Future Internet (AFI); Generic Autonomic Network Architecture; Part 2: An Architectural Reference Model for Autonomic Networking, Cognitive Networking and Self-Management, V1.1.1, May 2018 https://www.etsi.org/deliver/etsi_ts/103100_103199/10319502/01.01.01_60/ts_10319502v010101p.pdf

[CSPDT18] Mark Mortensen, John Abraham, Anil Rao, "CSP digital transformation: Leapfrogging competition via an Agile Transformation Framework", Analysss Mason White paper, February 2018 https://www-file.huawei.com/-/media/CORPORATE/PDF/News/CSP-Digital-Transformation-Leapfrogging-Competition-via-an-Agile-Transformation-Framework-Whitepaper.pdf?la=zh

[CY2020] CYNET OAV Architecture Analysis, Dec. 9, 2020 https://www.geant.org/Resources/Documents/GN4-3_White-Paper_CYNET_OAV_Architecture_Analysis.pdf

[D8818] Deliverable D8.8 Integrated Services Framework and Network Services Development Roadmap – Follow-Up, GÉANT Deliverable,December 14, 2018 https://www.geant.org/Projects/GEANT_Project_GN4/deliverables/D8-8_Integrated-Services-Framework-Roadmap-Follow-up.pdf

[DAIS20] I. Ioannou, V. Vassiliou, C. Christophorou and A. Pitsillides, "Distributed Artificial Intelligence Solution for D2D Communication in 5G Networks", in IEEE Systems Journal, doi: 10.1109/JSYST.2020.2979044 https://ieeexplore.ieee.org/document/9080591

[DFN20] DFN-GVS, Der General-Virtualization-Service des DFN, August 2020 https://www.win-labor.dfn.de/dfn-gvs/

[ENI20] Experiential Networked Intelligence (ENI) https://www.etsi.org/technologies/experiential-networked-intelligence

[ENSS20] ETSI OSM North Bound Interface API Description https://osm.etsi.org/wikipub/index.php/NBI_API_Description

[EOS20] European Open Science Cloud (EOSC) https://ec.europa.eu/research/openscience/index.cfm?pg=open-science-cloud

[EOSCP] EOSC Portal https://eosc-portal.eu/

[EOSM20] Open Source MANO website https://osm.etsi.org/

[ETS20] EOSC Technical Specification proposal, January 2020 https://confluence.egi.eu/display/EOSCDOC/Federation+services

[ETSIIFA022] ETSI GR NFV-IFA 022 V3.1.1 Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on Management and Connectivity for Multi-Site Services https://www.etsi.org/deliver/etsi_gr/NFV-IFA/001_099/022/03.01.01_60/gr_NFV-IFA022v030101p.pdf

[ETSIMANOA] ETSI MANO architecture (Whitestack) https://www.whitestack.com/products/whitenfv/

[ETSINFV002] ETSI GS NFV 002 V1.2.1 Network Functions Virtualisation (NFV); Architectural Framework https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf

[ETSINFVSPEC] ETSI standards for NFV: specifications within the NFV Architecture Framework https://www.etsi.org/technologies/nfv

[EZSM] ETSI GS ZSM 002 V1.1.1, Zero-touch network and Service Management (ZSM); Reference Architecture, August 2019 https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/002/01.01.01_60/gs_ZSM002v010101p.pdf

[EZSML] ETSI GR ZSM 004 V1.1.1, Zero-touch network and Service Management (ZSM); Landscape, March 2020 https://www.etsi.org/deliver/etsi_gr/ZSM/001_099/004/01.01.01_60/gr_ZSM004v010101p.pdf

[FAB19] K. Kinkade, ESnet a Key Partner on Project to Build Novel Network Research Infrastructure, Sept. 17, 2019 https://newscenter.lbl.gov/2019/09/17/esnet-partners-novel-network-research-infrastructure-project/

[FAIR18] EC, Turning FAIR Into Reality, Final Report and Action Plan from the European Commission Expert Group on FAIR Data, 2018 https://ec.europa.eu/info/sites/info/files/turning_fair_into_reality_1.pdf

[FED10] M. Campanella, et al., FEDERICA: A Virtualization Based Infrastructure for Future and Present Internet Research, International Conference on Testbeds and Research Infrastructures, TridentCom 2010: Testbeds and Research Infrastructures. Development of Networks and Communities pp 123-132 https://link.springer.com/chapter/10.1007/978-3-642-17851-1_9

[FGPPP14] View on 5G Architecture: 5G PPP Architecture Working Group, 2014 https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-5G-Architecture-WP-For-public-consultation.pdf

[FGPPP16] View on 5G Architecture: 5G PPP Architecture Working Group, v3.0, 2016 https://5g-ppp.eu/wp-content/uploads/2019/07/5G-PPP-5G-Architecture-White-Paper_v3.0_PublicConsultation.pdf

[FGPPP18] 5GPPP Architecture Working Group View on 5G Architecture, v2.0, Jan 2018 https://5g-ppp.eu/wp-content/uploads/2018/01/5G-PPP-5G-Architecture-White-Paper-Jan-2018-v2.0.pdf

[FGPPP20] 5GPPP Architecture Working Group View on 5G Architecture, v3.0, Feb 2020 https://5g-ppp.eu/wp-content/uploads/2020/02/5G-PPP-5G-Architecture-White-Paper_final.pdf

[FIR17] Future Internet Research and Experimentation (FIRE) https://web.archive.org/web/20170402025207/http://www.ict-fire.eu/

[GANA16] ETSI, GANA – Generic Autonomic Networking Architecture, White paper no. 16, October 2016 https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp16_gana_Ed1_20161011.pdf

[GEN18] GENI project https://www.geni.net/

[GNA20] Global Network Architecture, High-Level Architecture Plan https://gna-re.net/our-plan/

[GNA-G] Global Network Advancement Group https://www.gna-g.net/

[GOF14] GN3plus Project Achievements https://www.geant.org/Projects/GEANT_Project_GN4/Documents/GN3plus%20Year%201%20Project%20Achievements.pdf

[GRO18] GROOVY https://groovy-lang.org/

[GSUR19] D6.2 Automation and Orchestration of Services in the GÉANT Community, September 2019 https://www.geant.org/Projects/GEANT_Project_GN4-3/GN43_deliverables/D6-2_Automation-and-Orchestration-of-Services-in-the-GEANT-Community.pdf

[HAM13] J. van der Ham, F. Dijkstra, R. Lapacz, J. Zurawski, "Network Markup Language Base Schema version 1", OGF GFD-R-P.206, May 2013 https://www.ogf.org/documents/GFD.206.pdf

[HAZ14] M. Hazlinsky, B. Pietrzak, F. Farina, J. Sobieski, and P. Szegedi, SDNI: The GÉANT Testbeds Service - virtual network environments for advanced network and applications research," in 2014 First Int. Sc. and Tech. Conf. IEEE, Oct. 2014 https://ieeexplore.ieee.org/abstract/document/6995585/

[HDO17] J. S. Choi, "Hierarchical Distributed Orchestration Framework for Multi-Domain SDTNs", Journal of Optical Communication and Networking, vol. 9, no. 12, pp. 1125-1135, 2017 https://www.osapublishing.org/abstract.cfm?uri=jocn-9-12-1125

[HTA20] D10.4 – EOSC-hub technical architecture and standards roadmap, February 2020 https://www.eosc-hub.eu/deliverable/d104-eosc-hub-technical-architecture-and-standards-roadmap-v2-under-ec-review-0

[IBN18] SDx Central, What Is Intent-Based Networking?, June 2018 https://www.sdxcentral.com/networking/sdn/intent-based/definitions/what-is-intent-based-networking/

[IG14] Robert F. Smallwood, Information Governance: Concepts, Strategies, and Best Practices, Wiley, 2014 https://www.amazon.com/Information-Governance-Concepts-Strategies-Practices/dp/1118218302/ref=sr_1_1?dchild=1&keywords=9781118218303&linkCode=qs&qid=1586260980&s=books&sr=1-1

[LHC20] Large Hadron Collider / Compact Muon Solenoid (LHC/CMS) https://home.cern/about/experiments/cms

[MAL20] Gerben van Malenstein, AutoGOLE/SENSE WorkingGroup, GNA-G Community VCs, May 26, 2020 https://www.gna-g.net/wp-content/uploads/2020/06/5a-Gerben-van-Malenstein-GNA-G-AutoGOLE-SENSE-update.pdf

[MEF14] MEF 6.2, Carrier Ethernet Services Definition, August 2014 http://www.mef.net/Assets/Technical_Specifications/PDF/MEF_6.2.pdf

[MEF55] MEF Service Operations Specification, MEF 55, Lifecycle Service Orchestration (LSO): Reference Architecture and Framework, March 2016 https://www.mef.net/wp-content/uploads/2016/03/MEF-55.pdf

[MEFLSO] LSO Reference Architecture and Capabilities wiki https://wiki.mef.net/display/CESG/LSO+Reference+Architecture+and+Capabilities

[MON18] I. Monga, C. Guok, J. MacAuley, A. Sim, H. Newman, J. Balca, P. deMar, L. Winkler, T. Lehman, X. Yang, SDN for End-to-end Networked Science at the Exascale (SENSE), 2018 IEEE/ACM Innovation the Network for Data-Intensive Science (INDIS), Dallas, TX, USA https://ieeexplore.ieee.org/document/8648795/

[MRM20] Multi-Resource Markup Language (MRML) https://github.com/MAX-UMD/nml-mrml

[MSOS16] MEF Service Operations Specification MEF 55 Lifecycle Service Orchestration (LSO): Reference Architecture and Framework, March 2016 https://www.mef.net/wp-content/uploads/2016/03/MEF-55.pdf

[NBI20] SENSE Orchestrator North Bound API https://app.swaggerhub.com/apis/xi-yang/SENSE-O-Intent-API/2.0.2

[NFVPLREP] 4th ETSI NFV Plugtests Report V1.0.0 (2019-07) https://portal.etsi.org/Portals/0/TBpages/CTI/Docs/4th_ETSI_NFV_Plugtests_Report_v1.0.0.pdf

[NOV12] Leonidas Lymberopoulos, et al., NOVI Tools and Algorithms for Federating Virtualized Infrastructures, The Future Internet Assembly, FIA 2012: The Future Internet pp. 213–224 https://link.springer.com/chapter/10.1007/978-3-642-30241-1_19

[NSDT20] Network Service Description Template TOSCA specification https://openbaton.github.io/documentation/tosca-nsd/

[OB20] Open Baton platform http://openbaton.github.io/documentation/

[OBGH] Open Baton GitHub Source Code https://github.com/openbaton

[ODADC21] TM Forum Open Digital Architecture Concepts and Principles V2.1.0, March 2021 https://www.tmforum.org/resources/reference/gb998-open-digital-architecture-oda-concepts-principles-v2-1-0/

[ODAFA21] TM Forum, IG1167 ODA Functional Architecture Exploratory Report, v5.1.0, March 2021 https://www.tmforum.org/resources/exploratory-report/ig1167-oda-functional-architecture-exploratory-report-v5-1-0/

[ODAIG20] TM Forum, GB999 ODA Production Implementation Guidelines v4.0, April 2020 https://www.tmforum.org/resources/how-to-guide/gb999-oda-production-implementation-guidelines-v4-0/

[ODAW18] TM Forum Open Digital Architecture White paper, August 2018 https://www.tmforum.org/resources/whitepapers/open-digital-architecture/

[OGF20] Open Grid Forum (OGF) http://www.gridforum.org/

[ONAP20] ONAP Platform https://www.onap.org/

[ONAPB20] ONAP Use Cases and Blueprints https://www.onap.org/architecture/use-cases-blue-prints

[OSLICE] Openslice http://openslice.io/

[PIMM19] Product Inventory Management MEF 81, Requirements and Use Cases Reproduced with permission of the MEF Forum, November 2019 https://www.mef.net/wp-content/uploads/2019/11/MEF-81.pdf

[RESP] RESPOND-A Project https://respond-a-project.eu/

[SAT5G] SaT5G Project https://www.sat5g-project.eu/

[SEN20] SDN for End-to-end Networked Science at the Exascale, (SENSE) http://sense.es.net/

[SOB16] Jerry Sobieski, Milestone M6.1 Generalised Virtualisation Model v.2016, April 22, 2016 https://www.geant.org/Services/Connectivity_and_network/GTS/Documents/M6-1_Generalised-Virtualisation-Model-v2016_v02.pdf

[SPAW] Service Provider Architecture Wiki Page https://wiki.geant.org/display/NETDEV/SPA

[SSPM20] GN4-3, WP6 T2, D6.6 Transforming Services with Orchestration and Automation, November 2020 https://www.geant.org/Projects/GEANT_Project_GN4-3/GN43_deliverables/D6.6-Transforming_Services_with_Orchestration_and_Automation.pdf

[STA20] StackV Open Source Orchestration Software Suite http://github.com/MAX-UMD/StackV.community

[SUR2020] SURFnet OAV architecture analysis, Dec. 12, 2020 https://www.geant.org/Resources/Documents/GN4-3_White-Paper_SURFnet-OAV-Architecture-Analysis.pdf

[TADM11] TOGAF Architecture Development Methodology, 2011 https://pubs.opengroup.org/architecture/togaf91-doc/arch/chap05.html

[TAL19] H. Khalili, P. S. Khodashenas, D. Guija and S. Siddiqui, “Introducing Terrestrial Satellite Resource Orchestration Layer”, 21st International Conference on Transparent Optical Networks (ICTON), 2019, pp. 1–4, doi: 10.1109/ICTON.2019.8840562 https://ieeexplore.ieee.org/abstract/document/8840562/

[TGPPR14] Telecommunication management; Management concept, architecture and requirements for mobile networks that include virtualized network functions, 3GPP specification 128.500, version 14.1.0 Release 14 https://www.etsi.org/deliver/etsi_ts/128500_128599/128500/14.01.00_60/ts_128500v140100p.pdf

[TNMS20] TotalNMS – Total Network Management System to support dynamic business needs https://www.gilat.com/technology/totalnms/

[VAL19] B. Valera-Muros, P. Merino, Is GÉANT Testbeds Service compliant with ETSI MANO?, Sept. 2019, available from DOI: 10.1109/5GWF.2019.8911622 https://ieeexplore.ieee.org/abstract/document/8911622/

Glossary

3GPP	3rd Generation Partnership Project
5G PPP	5G Infrastructure Public Private Partnership
A&AI	Active and Available Inventory
AA	Authentication and Authorisation
AAI	Authentication and Authorisation Infrastructure
AE	Autoscaling Engine
AFI	Autonomic network engineering for the self-managing Future Internet
AI	Artificial Intelligence
AMC	Automated and Autonomic Management & Control
API	Application Programming Interface
APPC	Application Controller
AutoGOLE	Automated Open Lightpath Exchange
AWA	Application Workflow Agent
AWS	Amazon Web Services
BDix	Belief-Desire-Intention intelligent agent with extended capabilities
BMS	Bare Metal Server
BPF	Business Process Framework
BSS	Business Support System
BUS	Business Application
CCVPN	Cross Domain and Cross Layer VPN
CD	Continuous Development
CDT	Controller Design Tool
CFS	Customer-Facing Service
CI	Continuous Integration
CIA	Confidentiality, Integrity and Availability
CLAMP	Control Loop Automation Management Platform
CLI	Command Line Interface
CMS	Compact Muon Solenoid
CNF	Container Network Function
CPE	Customer Premises Equipment
CRM	Customer Relationship Management
CRUD	Create, Retrieve, Update, Delete
CSF	Customer-Facing Service
CUS	Customer Application Coordinator
D2D	Device to Device
DAI	Distributed Artificial Intelligence
DAIS	Distributed Artificial Intelligence Solution
DC	Data Centre

DCAE	Data Collection, Analytics and Events
DCSP	Data Centre Service Provider
DDOS	Distributed Denial of Service
DE	Decision-making Element
DevOps	Development and Operations
DM	Domain Manager
DNS	Domain Name System
DSC	Digital Service Customer
DSL	Domain Specific Language
DSP	Digital Service Provider
DTN	Data Transfer Node
E-W	East-West
E2E	End to End
ECM	Element Control and Management
ECOMP	Enhanced Control, Orchestration, Management and Policy
EM	Element Manager
EMS	Element Management System
ENI	Experiential Networked Intelligence
EOSC	European Open Science Cloud
eTOM	enhanced Telecom Operations Map
ETSI	European Telecommunications Standards Institute
EU	European Union
FAIR	Findable, Accessible, Interoperable, and Reusable
FEDERICA	Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures
FIRE	Future Internet Research and Experimentation
FM	Fault Management
GANA	Generic Autonomic Network Architecture
GCS	GÉANT Connection Service
GENI	Global Environment for Network Innovations
GNA	Global Network Architecture
GOFF	GÉANT Open Flow Facility
GTS	GÉANT Testbed Service
GUI	Graphical User Interface
GVM	Generalized Virtualization Model
GVS	Generalized Virtualization Service
GXP	Global Exchange Point
HNF	Hybrid Network Function
HPC	High-Performance Computing
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IC	Infrastructure SDN Controller
ICM	Infrastructure Control and Management
ID	Identity
IETF	Internet Engineering Task Force
InP	Infrastructure Provider
IoT	Internet of Things
IP	Internet Protocol

ISG	Industry Specification Group
JSON	JavaScript Object Notation
KDU	Container Deployment Unit
KP	Knowledge Plane
KPI	Key Performance Indicator
LCM	LifeCycle Manager
LHC	Large Hadron Collider
LSO	Lifecycle Service Orchestration
MAN	Metropolitan Area Network
MANO	Management and Orchestration
MBTS	Model-Based Translation Service
MCE	Model Computation Element
ME	Managed Entity
MEC	Mobile Edge Computing
MEC	Multi-Access Edge Computing
MEF	Metro Ethernet Forum
MIB	Management Information Base
MNO	Mobile Network Operator
MON	Monitoring Module
MPLS	Multiprotocol Label Switching
MQ	Message Queue
MRML	Multi-Resource Markup Language
MSP	Managed Service Provider
MTA	Multi-Tenancy Application
MVNO	Mobile Virtual Network Operator
NaaS	Network as a Service
NBI	Northbound Interface
NetDevOps	Network Development and Operations
NFV	Network Function Virtualisation
NFVI	Network Function Virtualisation Infrastructure
NFVO	Network Function Virtualisation Orchestrator
NGMN	Next-Generation Mobile Networks
NM	Network Management
NML	Network Markup Language
NMS	Network Management System
NOC	Network Operations Centre
NOP	Network Operator
NOVI	Network Innovation over Virtualised Infrastructures
NREN	National Research and Education Network
NS	Network Service
NSaaS	Network Slice as a Service
NSD	Network Service Descriptor
NSE	Network Slicing Engine
NSF	National Science Foundation
NSI	Network Slice Instance
NSI	Network Service Interface
NSMF	Network Slice Management Function
NSR	Network Service Record

NSSMF	Network Sub- Slice Management Function
NST	Network Slice Template
NTECH	Network Technologies
OASIS	Organisation for the Advancement of Structured Information Standards
OAV	Orchestration, Automation and Virtualisation
ODA	Open Digital Architecture
ONAP	Open Networking Automation Platform
ONF	Open Networking Foundation
ONIX	Overlay Network for Information eXchange
OOM	ONAP Operations Manager
OSM	Open Source MANO
OSS	Operations Support System
OTRS	Open-Source Ticket Request System
OTT	Over-the-top
PA	Provider Agent
PDU	Physical Deployment Unit
PNF	Physical Network Function
PoC	Proof of Concept
POL	Policy Module
PoP	Point of Presence
PPP	Public Private Partnership
QoS	Quality of Service
R&E	Research and Education
RAN	Radio Access Network
RCA	Resource Control Agent
RDB	Resource Database
RDF	Resource Description Framework
REST	Representational State Transfer
RF	Resource Function
RFS	Resource-Facing Service
RM	Resource Manager
RO	Resource Orchestration
RPC	Remote Procedure Call
RT	Resource Tree
RTIR	Request Tracker for Incident Response
SAML	Security Assertation Markup Language
SaT5G	Satellite and Terrestrial Network for 5G
SC	Service Customer
SDC	Service Design and Creation
SDK	Software Development Kit
SDM-C	Software-Defined Mobile Network Controller
SDM-O	Software-Defined Mobile Network Orchestrator
SDM-X	Software-Defined Mobile Network Coordinator
SDN	Software-Defined Network
SDNC	SDN Controller
SDP	Service Demarkation Point
SDTN	Software-Defined Transport Network
SENSE	SDN for End-to-end Networked Science at the Exascale

SENSE-O	SENSE Orchestrator
SF	Service Function
SFA	Slice Federated Architecture
SFC	Service Function Chaining
SFCO	SFC Orchestrator
SID	Shared Information Model
SLA	Service-Level Agreement
SO	Service Orchestrator
SOF	Service Orchestration Functionality
SON	Self-Organising Network
SP	Service Provider
SPA	Service Provider Architecture
SSH	Secure Shell
SSP	Self-Service Portal
STP	Service Termination Point
T	Task
TALENT	Terrestrial Satellite Resource Coordinator
TAM	The Application Framework
TAPI	ONF Open Transport API
TC	Tenant SDN Controller
TMF	Tele Management Forum, TM Forum
TOGAF	The Open Group Architecture Framework
TOSCA	Topology and Orchestration Specification for Cloud Applications
TSDB	Time Series DataBase
UI	User Interface
URL	Uniform Resource Locator
URLLC	Ultra Reliable Low Latency Communication
UUI	Use Case UI
VC	Virtual Circuit
VCA	VNF Configuration and Abstraction
VCA	VNF Configuration and Abstraction Module
VDU	Virtual Deployment Unit
VFC	Virtual Function Controller
VI	Virtual Infrastructure
VID	Virtual Infrastructure Deployment
VIM	Virtual [or Virtualised] Infrastructure Manager
VISP	Virtualisation Infrastructure Service Provider
VM	Virtual Machine
VNF	Virtual Network Function
VNF-FG	VNF Forwarding Graph
VNFC	VNF Controller
VNFD	VNF Descriptor
VNFM	Virtual Network Function Manager
VNFR	VNF Record
VoLTE	Voice over Long-Term Evolution
VPN	Virtual Private Network
VSI	Virtual Switch Instance
VSP	Virtual Software Product

WAN	Wide Area Network
WIM	WAN Infrastructure Manager
WP	Work Package
WP6	Work Package 6 Network Technologies and Services Development
WP6 T2	WP6 Task 2 Network Services Evolution and Development
YAML	Yet Another Markup Language
ZSM	Zero-touch network and Service Management