09-12-2020

SURFnet OAV Architecture Analysis

Grant Agreement No.:	856726
Work Package	WP6
Task Item:	Task 2
Dissemination Level:	PU (Public)
Document ID:	GN4-3-20-274ffd5

Authors: Sonja Filiposka (UKIM/MARnet), Susanne Naegele Jackson (FAU/DFN), Maria Isabel Gandia Carriedo (CSUC/RedIRIS), Donal Cunningham (HEANET), Eduardo Jacob (RedIRIS), Hamzeh Khalili (I2CAT/RedIRIS), Iacovos Ioannou (CYNET), Ivana Golub (PSNC), Jasone Astorga (RedIRIS), Kostas Stamos (GRNET), Martin Dunmore (Jisc), Roman Lapacz (PSNC), Simone Spinelli (GEANT), 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

This document analyses the mapping of the SURFnet architecture to the TM Forum's Open Digital Architecture, as part of the GN4-3 project's ongoing activity in the Network Technologies and Service Development's work package to provide a standardised view of the components and implementations of orchestration, automation and virtualisation within the NRENs. Overall, it can be concluded that the SURFnet OAV architecture, although independently developed, is very much aligned to the ODA's core principles and design concepts.

Table of Contents

Table of Figures

Executive Summary

Using a common reference architecture to analyse NREN architectures from an orchestration, automation and virtualisation (OAV) point of view helps align the NRENs' efforts and find commonalities in the way they implement different functionalities and components. To this end, the GN4-3 WP6 Network Technologies and Services Development OAV team has selected the TM Forum Open Digital Architecture (ODA) to be used as a reference blueprint architecture for cross-comparison.

This document presents an analysis performed by the WP6 OAV team of the different functional aspects of the SURFnet network architecture, mapping its components to the TM Forum ODA. The mapping highlights the main characteristics and functionalities of the current SURFnet network architecture and how they fit into the main functional domains of ODA.

The analysis has been carried out based on published information about the architecture which has been confirmed and augmented based on direct consultation with the SURF network architects. Overall, it can be concluded that the SURFnet OAV architecture, although independently developed, is very much aligned to ODA core principles and design concepts.

1 Introduction

As part of its activities towards driving greater adoption of OAV principles in the NRENs, the Orchestration, Automation and Virtualisation (OAV) team within the GN4-3 Network Technologies and Services Development Work Package (WP6) has selected the TM Forum Open Digital Architecture [ODA] as a reference architecture. The rationale for its selection, and more information about ODA, are given in [D6.6].

To help establish a common context for the whole NREN community for their work in orchestration, automation and virtualisation, the WP6 OAV team is offering to perform the mapping of OAV architectures of individual NRENs to the TM Forum ODA design principles and concepts. One example of such a mapping has already been published [CYNET], and several more are in progress.

This white paper analyses the current OAV architecture of SURFnet, the Dutch education and research network. SURF is one of the NRENs in the GÉANT community that has been focusing for some time on improving and advancing its network architecture and management by introducing automation and orchestration in all its workflows related to network services while working on reconstructing their SURFnet network at the hardware level.

The analysis was carried out based on the most recent information available (at the end of 2019) [AUTO], [ARCH], [DATP] as well as through discussion with SURF. The authors would in particular like to thank Migiel de Vos and Jac Kloots of SURF for their comments and assistance in producing this white paper.

Section 2 of this document provides a short overview of the TM Forum ODA. Section 3 explains the mapping, while conclusions are summarised in Section 4.

Further information on the OAV work being performed in GN4-3 WP6 can be found in the OAV Wiki at [OAVWIKI], including links to events, a community portal showing OAV work in the NRENs, and OAV training material.

2 TM Forum Open Digital Architecture

The TM Forum established its ODA as a blueprint for new digital industry architectures with corresponding common terminology and a well-defined set of core design principles. The ODA provides a reference architecture that maps the main functionalities in its functional blocks and enables orchestration and automated operations. The main idea behind ODA is based on decoupling and integration of components which enable an independent choice of solutions for each component, while at the same time maintaining a unified overall approach that supports the full end-to-end service lifecycle, including interoperability. The high-level ODA functional architecture represents the main component capability-based grouping into the so-called ODA functional blocks (see Figure 2.1).

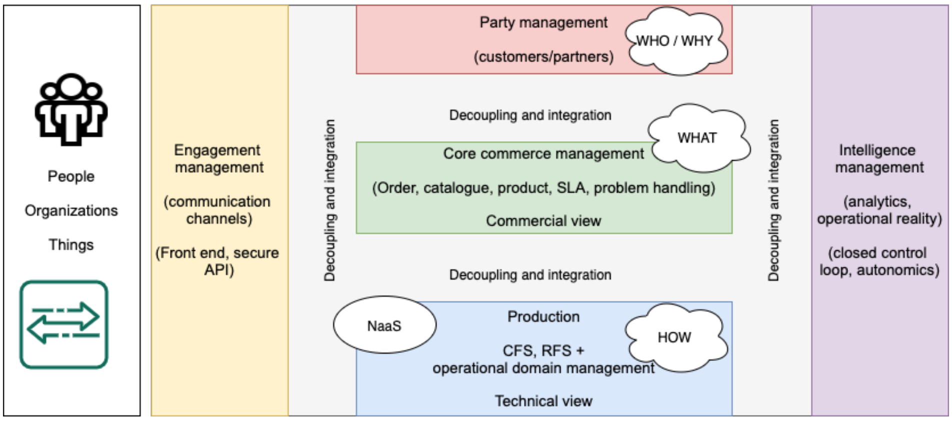
People
Organizations
Things
Engagement management
(communication channels)
(Front end, secure API)
Decoupling and integration
Party management
(customers/partners)
WHO / WHY
Decoupling and integration
WHAT
Core commerce management
(Order, catalogue, product, SLA, problem handling)
Commercial view
Decoupling and integration
NaaS
Production
CFS, RFS +
operational domain management
HOW
Technical view
Decoupling and integration
Intelligence management
(analytics, operational reality)
(closed control loop, autonomics)

Figure 2.1: The TM Forum ODA functional architecture

In a nutshell, the Engagement Management functional block focuses on the engagement with the end-users (people and systems) that can interact via multiple possible channels; the Party Management functional block handles the processes that are related to all parties that interact with the organisation and defines their roles and relationships; the Intelligence Management functional block is in charge of the implementation of data analytics processes and, based on the analysis carried out, provides closed control loops for full automation wherever possible; the Core Commerce functional block is focused on the placement of products and services to the customers and manages the product lifecycle; finally, the Production functional block manages the delivery and lifecycle of all customer-facing and resource-facing services that can be based on different technologies or might be a combination of multiple operational domains, including multi-domain services provided with the cooperation of other parties.

3 Architecture Analysis

SURF is going through a major upgrade of its network architecture moving from SURFnet7 to SURFnet8. The new architecture relies on a full mesh topology between routers, using MPLS with segment routing that allows for very fast rerouting. There are many changes involved in this upgrade, however the analysis provided here only focuses on its orchestration and automation aspects and related information.

3.1 Goals and Requirements

One of the premises of SURFnet8 is that the complete service provisioning and management are performed exclusively via predefined workflows. This means that CLI access to network devices should no longer be used by network engineers. Currently, this goal has not been implemented for all services, as the migration is still ongoing. This migration project is quite large and designed to take place in phases, taking into account that the scope of the SURFnet8 service layer includes over 280 locations, 400 networking devices and 630 services [AUTO].

The main driver for introducing automation and orchestration in SURFnet was the focus on the customer. The benefits of OAV are seen not only in achieving Configuration Management Database (CMDB) integrity and accurate monitoring, which are relevant for NOC teams, but also in its ability to provide predictable service delivery to customers in shorter time spans. By automating processes such as service provisioning, self-service for the users can be achieved. Self-service that transfers control to the customer is considered another positive impact of the implementation of OAV in addition to the improved management of the complete service lifecycle (provisioning, change-add-remove, termination) and the ability to offer composable services.

The SURFnet automation team numbers around 10 people with a software engineering background including testers. The software development is implemented using Agile methodologies with each sprint lasting 2 weeks.

The main requirements for the implementation of orchestration on top of automation defined by the SURFnet development team include:

- Single point of truth – the information about the network stored in the inventory is considered to be the correct/intended information at any point in time. If the network does not reflect the information in the inventory, then it needs to be reconfigured so that it corresponds to the inventory.
- Solid information model – consistent modelling of resources and services across components and interfaces.

- Decoupling and well-defined interfaces – all components are accessed via well-defined REST APIs based on the shared information model.
- Automated administration – network engineers use the same approach as customers when they need to add/change network configuration.
- Predefined processes for service provisioning – a workflow is defined with all the standardised steps necessary to deploy each specific service type.

3.2 Technology Domains

The final goal envisioned is to move towards working with different technology domains. All networking devices that belong to one technology domain will be managed uniformly, thus enabling the creation of standardised workflows at the orchestration layer. The current technology domains included in the SURFnet architecture are:

- The Service domain – with Juniper MX routers as network resources
- The Network Function Virtualisation (NFV) domain
- The Wireless domain
- The Optical domain – moving towards ECI Apollo

3.3 High-Level Architecture Overview

The complete solution is decoupled into a mix of commercial, open-source and in-house components. A separate API is developed in-house for interaction with all customers via the dashboard GUI-based interface where AAI control is enforced. The high-level architecture as illustrated by SURF is presented in Figure 3.1.

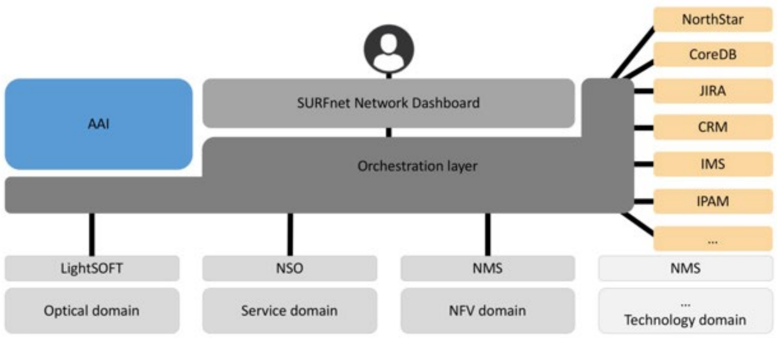

Figure 3.1: SURFnet OAV architecture as provided in [AUTO]

3.4 Mapping to the ODA Functional Architecture

When placed into the context of the TM Forum ODA functional representation, the SURFnet OAV architecture can be represented as shown in Figure 3.2. The grey boxes in the diagram represent SURFnet architecture components, and their placement within the ODA functional blocks is defined based on their main functionalities.

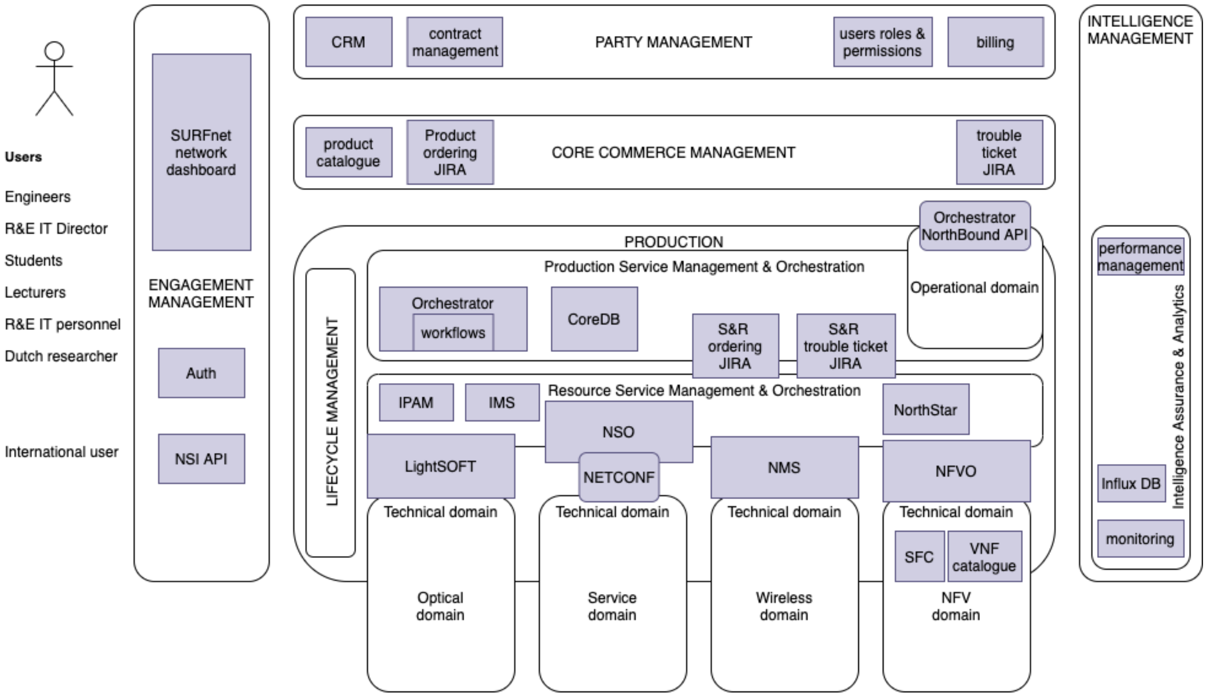

Figure 3.2: SURFnet OAV architecture mapped to the TM Forum ODA

When compared to the ODA core concepts and design principles, the SURFnet architecture has considered and adopted a large number of very similar ideas and approaches. The architecture is designed as a loosely coupled system with a number of standalone components that exchange information using well-defined APIs and a consistent information model.

3.4.1 Engagement Management

Components such as the SURFnet network dashboard and the separate Authentication module (part of the AAI implementation), which are part of the Engagement Management functional block in the ODA, serve the main engagement functionalities, such as the interaction with the external and internal users, the channels used for interaction (in this case the front-end GUI and NSI API), authentication and authorisation, and the definition of the customer journey based on the context. The SURFnet Network Dashboard is developed in such a way that all logic that needs to be implemented as a response to user interaction is based on the rest of the modules in the corresponding functional blocks.

In this way, a "standardised" API is exposed to the partners and any external systems while implementing access management. All users (including the network engineers) use this engagement module to interact with the system, each according to their specific role-based rights and privileges defined in the Party Management modules. This uniformity in interaction enables a single point of contact that needs to be managed and enforces a standardised way of interaction that is easier to control when implementing orchestration and automation. In this way, it is possible to ensure that there will be no "shortcuts" implemented and that all workflows will be followed by all actors.

3.4.2 Party Management

The information about all users is held in a single Customer Relationship Management (CRM) system placed in the Party Management functional block. The CRM system is responsible for handling information about people and organisations, including partners and employees. By having one single place in which to store and manage all "party" information, data integrity is increased and the "service silo" mentality is broken, escaping the notion of a separate user database for each service. The decision about which type of "party" can manage which type of service/resource are made based on role-based rules cross-referencing the information defined in the CRM system and the product/service catalogue.

This decoupling of party management-related functions enables easier implementation of any billing for the services used if required. The component that manages billing may need input on product/service usage, depending on the contracts and the rules on billing for a particular product.

In addition, within the party management functional block, another module is mapped that manages the lifecycle of contracts with users/organisations, but also potentially partners and other types of parties. The contracts define the subset of views that are provided to the specific party when considering the Core Commerce Management functional blocks, such as a subset of the products offered depending on existing contracts.

3.4.3 Core Commerce Management

The product catalogue and product order management functionalities implemented in SURFnet are placed within the Core Commerce Management and Orchestration functional block. The available documentation about the SURFnet architecture lists the products that are available (or planned to be available) to external and internal customers as:

- Products available to external customers:
  - Service ports
  - IP service
  - Lightpath (EPL)
  - ELAN service
  - L3VPN service
- Products available to internal customers (NOC-facing services):
  - Nodes
  - Core links
  - IP peering
  - IPv4/IPv6 prefix (customer prefix administration)

The product catalogue as a separate component with all its functionalities is currently still being developed but is recognised as being a vital component of the overall system.

The product catalogue should also include configuration management information, i.e. the definition of the underlying components needed to provide the product to the customer. In the SURFnet architecture terms, this includes the mapping between a product and its necessary product blocks, including any additional product configuration (product-related input variables or specified workflows and other elements that are considered essential to manage the product). This formalised way of describing the underlying components of a product enables flexible composition of services and/or resources. As presented in Figure 3.3, the SURFnet approach is very similar to the composable approach as defined in the TM Forum Shared Information Data (SID) which includes products, customer-facing services, resource-facing services and resources.

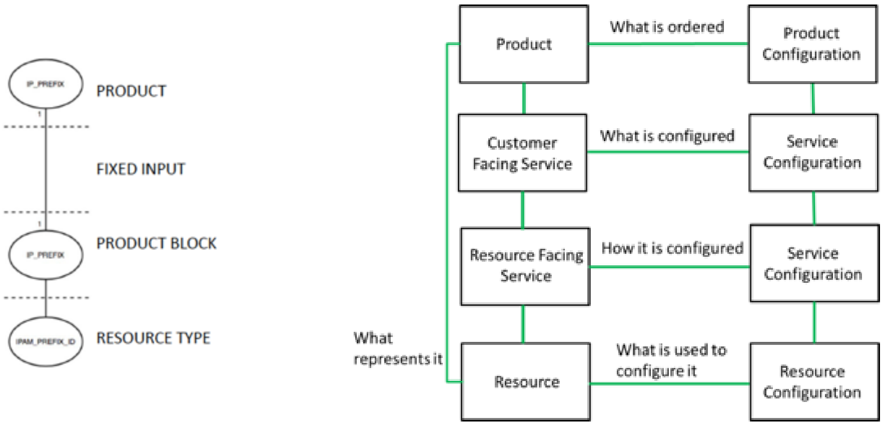

Figure 3.3: SURFnet product decomposition [AUTO] vs TMForum SID product decomposition based on [GB922]

Jira is used as a ticketing system that tracks the status of all orders from creation until completion. Although the main SURFnet "orchestrator" component is placed in the production functional block, there may be a number of very high-level workflows defined that deal with the product ordering and decomposition into required services and resources for the specific product that should be placed in the core management functional block. In this mapping, the orchestration component has been placed in the production functional block, aiming to show that the majority of defined workflows are specifying the steps necessary to manage the lifecycle of each service.

Jira is also used to manage the Trouble Ticketing functionalities. The highest level of trouble tickets is on a product level, and they may be correspondingly linked to relevant trouble tickets related to services and/or resources that are part of the product. The product level trouble tickets enable an integrated view for the users, where the information about the status of these tickets is presented via the dashboard front-end.

At the moment, there is no Service Level Agreement (SLA) component that can be mapped to this functional block, but the modular design supports the future integration of an SLA management component if necessary.

3.4.4 Production

The main logic of the architecture is placed in the production functional block. This block is the one that is tasked with delivering the lifecycle of the Customer-Facing Services (CFSs) and Resource Facing Services (RFSs). The ODA definition embraces the concept that this functional block should expose a single API towards the rest of the environment, aka the NaaS API in the ODA documentation. Similarly, the SURFnet architecture is based on the orchestrator API that exposes all functionalities of the underlying component as it is presented in Figure 3.4.

The SURFnet orchestrator performs end-to-end service management, focusing on the application of policy control and service performance and quality. One of the main supporting components to enable the orchestrator to perform the defined workflows that manage the services and resources in their different lifecycle stages is the service inventory or CoreDB in SURFnet. This component stores information about all service instances and their current configuration. Workflows are defined not only for service provisioning but also for testing and monitoring, providing all aspects of service fulfilment. They also have incorporated error response mechanisms that make sure that the network and related systems will always be in a stable state even in the case where an error occurs and the process instance does not go as planned. The aim is for all standardised services to be automated and orchestrated in this way. For the custom bespoke services, manual implementation is used.

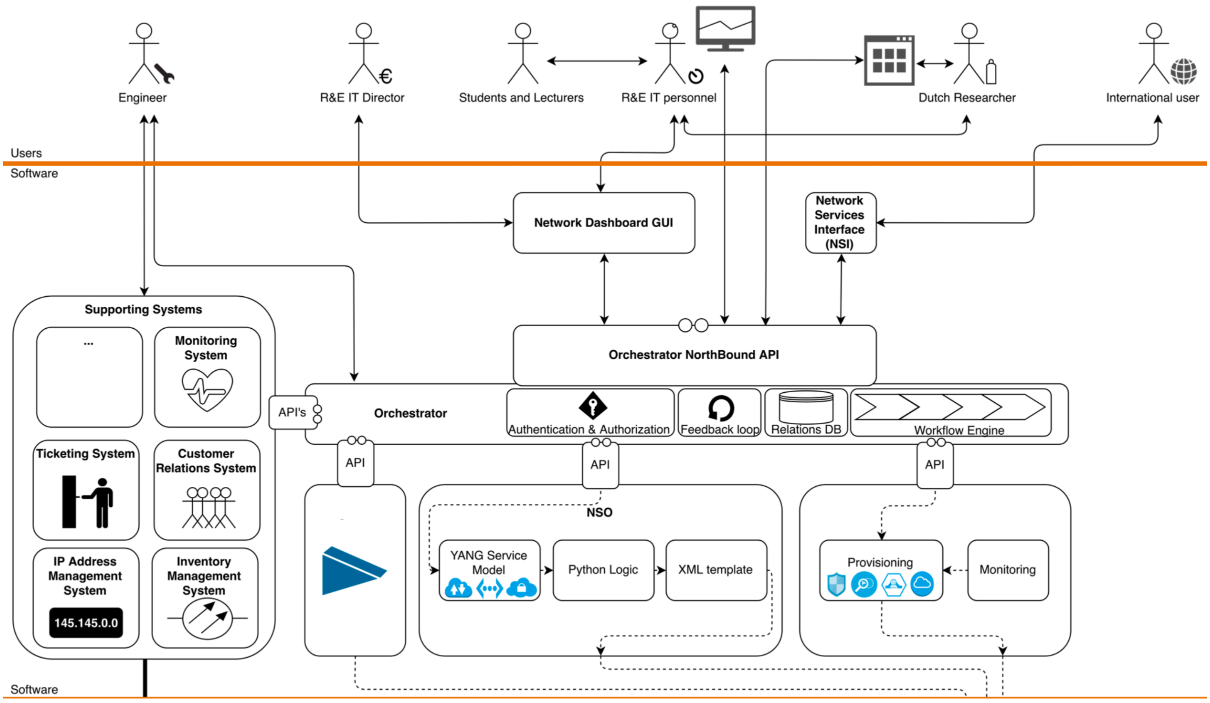

Figure 3.4: Orchestrator API as a single point of interaction [AUTO]

In addition to CoreDB, on the resource orchestration level, one of the main components is the inventory management system (IMS) which is used to store information about available resources (and fibre services). Lower level workflows of the orchestrator handle resource-related processes, such as adding a new router. IP address management (IPAM) component is also mapped within resource management as a type of component that manages the overall usage of IP addresses as a specific type of resource in the network. The North Star component that is also mapped to resource management is a path calculation element that is currently being used for flow visualisation only.

Across both service and resource orchestration, there are corresponding service and resource ordering tickets as well as service and resource trouble tickets, all managed via Jira. As previously mentioned they are linked to corresponding product orders and product trouble tickets whenever a user is affected by these activities.

It should be noted that service development and testing also belong within this production service and management orchestration block of the production functionalities. For these purposes, in SURFnet the development, testing, acceptance, production (DTAP) approach is used with GitLab and CI/CD pipelines. The staging and development environment is created using a Virtual Juniper MX testbed, where behaviour-driven testing is used to check if the devices behave as expected. Based on the available documentation, this approach is currently in use for services that are provided within the "service" technology domain only.

3.4.5 Technical Domains

As previously discussed, there are several identified technology domains in SURFnet, and each of these has its own resource service and management orchestrator, such as LightSOFT for the optical technical domain. This section describes the service and NFV technical domains in more detail since they are both currently being implemented using OAV techniques. The optical and wireless domains will be automated as needed in future generations of the SURFnet architecture.

3.4.5.1 *"Service" Technical Domain*

The most advanced domain in terms of automation and orchestration is the so-called "service" technical domain, based on an all-MPLS network, for which the resource orchestrator of choice is Cisco NSO. Each resource node in this domain (all nodes are Juniper MXs) has an equal role/function, and the northbound API for each node is NETCONF. This implementation is presented in Figure 3.5.

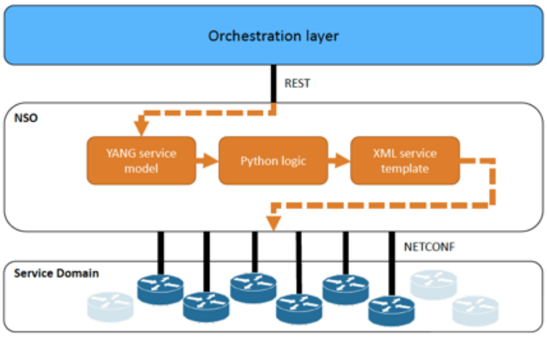

Figure 3.5: Resource service management and orchestration using NSO in SURFnet [AUTO]

Using ODA terminology, the RFS services are modelled in NSO by extending the existing basic YANG service model provided by NSO. For more complex services additional Python scripts can be run on the YANG model to improve validation and complex service description. Based on the service YANG model and its mapping to the devices where the service should be configured, the corresponding XML service template is produced for a generic NETCONF NED (NSO network element driver). Based on the XML template, the new device configuration is compiled for each device. The service is configured on the chosen devices by pushing the new device configuration using NETCONF. In this way, all NETCONF options such as roll-back on error, or network-wide transactions are available for use and all underlying devices are treated consistently. In addition, NSO has the ability to oversee the pushed configuration to the devices and report any cases where the configuration has been changed by a third party, thus adding another layer of configuration integrity assurance. It should be noted that the most “software programming aspect” of using NSO is the definition of the YANG service model, which is further made easier by available templates, but requires good design skills and YANG modelling knowledge. Thus, it may be considered that the main software efforts in the overall solution are required in the development of the orchestrator and workflows.

3.4.5.2 *NFV Technical Domain*

The NFV technical domain is based on the ETSI NFV MANO specification [MANO]. The NFV orchestrator (NFVO) is used to orchestrate resources and provide NFV-based services that can then be combined with services from other technical domains. The main examples of services that are added to the service portfolio include firewall as a service in two flavours: public traffic firewalling and customer circuits firewalls. vRouter and vEPC are also on the list of offered services.

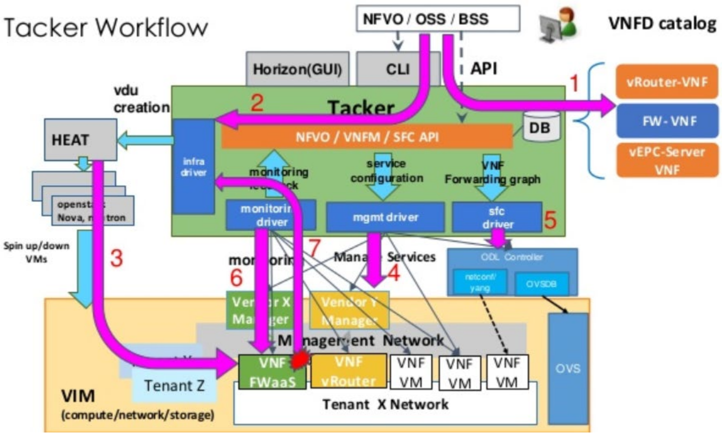

Figure 3.6: Resource service management and orchestration in the NFV technical domain [SNFV]

The whole NFV technical domain is built on top of OpenStack, where the OpenStack Tacker component is used to manage the VNF lifecycle playing the role of NFVO and a generic VNF Manager (VNFM), as shown in Figure 3.6. It offers its own GUI, the Horizon, and can also be accessed via CLI and API. The NFVO is responsible for end-to-end deployment of all services requested from the NFV technical domain by decomposing them into VNFs. The VNFM is in charge of the lifecycle of each separate VNF. TOSCA is used for VNF metadata definition. The Tacker service function chaining (SFC) feature is used to chain multiple VNFs. The SFC rules are pushed to the virtual infrastructure and the Virtual Packet Processing (VPP) virtual switches on the computing nodes via the OpenStack Neutron component which provides the network as a service functionality. Cisco open-source VPP is used as a virtual switch instead of the usual OVS instances.

3.4.6 Intelligence Management

The final functional block focuses on Intelligence Management. This includes any data stored for performance management and other trend-related types of analysis, usually using Big Data analytics due to the nature of the gathered data. Currently, the SURFnet architecture implements components mapping to this functional block that focus on data gathering and data storing functionalities, such as monitoring components and stored monitoring data in InfluxDB. Performance management components that are based on the analysis of this stored data are also mapped to this functional block.

The team has also implemented some advanced data analytics using Machine Learning (ML) algorithms such as traffic prediction for capacity management.

Currently, all activities related to service assurance, that is troubleshooting and solving incidents/problems reported in trouble tickets, are being performed manually by the NOC staff. Future enhancements of the architecture might include automated control loops that will heal the network in case of recognised issues that have “standard” solution procedures. These automated control loops will be based on the knowledge stored and analysed in this functional block, while their actions will be reflected in related workflows in the orchestrator mapped in the Production block.

4 Conclusions

Overall, it can be concluded that the SURFnet OAV architecture, although independently developed, is very much aligned to the ODA core principles and design concepts and supports a strong argument for promoting the usage of automated and orchestrated horizontal solutions. The approach adopted is iterative, scalable and largely modular, and the orchestrator is implemented in a technology-agnostic manner. The information governance process is also very well designed, while the API-based building blocks enable integration with other external systems in the organisation when needed. When considering multi-domain services as a use-case, the single API that is used to communicate with the orchestrator enables the SURFnet domain to be accessible to partners from the outside, while still allowing control of which and how many of the network capabilities to expose to the outside world.

References

[ODA] TMForum Open Digital Architecture White paper, 01-08-2018

[D6.6] Deliverable D6.6 Transforming Services with Orchestration and Automation

[CYNET] CYNET OAV Architecture Analysis

[AUTO] Sadi Koçak, Automation @ SURFnet, 10th SIG-NOC meeting, 14-11-2019

[ARCH] Jac Kloots, SURFnet NREN Network Architecture, Internet2 meeting 07-05-2018

[DATP] Jac Kloots, DATP@SURFnet, 1st NGN meeting, 22-11-2018

[OAVWIKI] https://wiki.geant.org/display/OAV

[GB922] TMForum GB922 Configuration R14.5 – Configuration and Profiling Business Entity Definitions

[MANO] ETSI NFV MANO specification

[SNFV] Eyle Brinkhuis, The journey towards an NFV-infrastructure, 19th STF, 24-02-2020

Glossary

AAI	Authentication and Authorisation Infrastructure
API	Application Programming Interface
CFS	Customer-Facing Service
CI/CD	Continuous Integration / Continuous Delivery
CLI	Command Line Interface
CMDB	Configuration Management Database
CRM	Customer Relationship Management
DTAP	Development, Testing, Acceptance, Production
ELAN	Ethernet Local Area Network
GUI	Graphical user interface
IMS	Inventory Management System
IP	Internet Protocol
IPAM	IP address management
L3VPN	Layer 3 Virtual Private Network
ML	Machine Learning
MPLS	Multiprotocol Label Switching
NaaS	Network as a Service
NED	Network Element Driver
NETCONF	Network Configuration Protocol
NFV	Network Function Virtualisation
NFVO	Network Function Virtualisation Orchestrator
NOC	Network Operations Centre
NSO	Network Service Orchestrator
OAV	Orchestration, Automation and Virtualisation
ODA	Open Digital Architecture
OVS	Open Virtual Switch
RFS	Resource-Facing Service
SID	Shared Information Data
SLA	Service Level Agreement
TOSCA	Topology and Orchestration Specification for Cloud Applications
VNF	Virtual Network Function
VNFM	Virtual Network Function Manager
VPP	Vector Packet Processing
YANG	Yet Another Next Generation