TRANSACTIONS ON EMERGING TELECOMMUNICATIONS TECHNOLOGIES
Trans. Emerging Tel. Tech. 2016; **27**:1206–1215
Published online 25 July 2016 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/ett.3084

SPECIAL ISSUE ARTICLE

DevOps for network function virtualisation: an architectural approach

Holger Karl[1], Sevil Dräxler[1], Manuel Peuster[1], Alex Galis[2], Michael Bredel[3], Aurora Ramos[4]*, Josep Martrat[4], Muhammad Shuaib Siddiqui[5], Steven van Rossem[6], Wouter Tavernier[6] and George Xilouris[7]

[1] Paderborn University, Paderborn, Germany
[2] University College London, London, UK
[3] NEC Laboratories Europe, Heidelberg, Germany
[4] ATOS Spain, Madrid, Spain
[5] Fundació i2CAT, Barcelona, Spain
[6] Ghent University – iMinds, INTEC, Gehnt, Belgium
[7] Institute of Informatics and Telecommunications, NCSR 'Demokritos', Greece

ABSTRACT

The Service Programming and Orchestration for Virtualised Software Networks (SONATA) project targets both the flexible programmability of software networks and the optimisation of their deployments by means of integrating Development and Operations in order to accelerate industry adoption of software networks and reduce time-to-market for networked services. SONATA supports network function chaining and orchestration, making service platforms modular and easier to customise to the needs of different service providers, and introduces a specialised Development and Operations model for supporting developers. © 2016 The Authors. *Transactions on Emerging Telecommunications Technologies* published by John Wiley & Sons, Ltd.

***Correspondence**
Aurora Ramos, ATOS Spain, Madrid, Spain.
E-mail: aurora.ramos@atos.net

Received 6 April 2016; Revised 14 June 2016; Accepted 27 June 2016

1. INTRODUCTION

In the ongoing research discussion about the structure and architecture of 5G networks, a basic trend is evident: the network will have to be much more flexible and open to profound changes of its operation to match evolving demands. These demands will not just evolve quantitatively but also qualitatively in that traffic with different, yet unforeseeable, characteristics will have to be supported. An ideal approach to increase flexibility of a system is to scale software, more precisely, of reconfigurable and replaceable software. This is the core notion of providing conventionally fixed network functions via software, typically in a virtualised environment: network function virtualisation (NFV) [1]. It is a transformation that relies on software-based innovation on top of more generic telecommunications hardware.

This disruptive architecture is challenging to put into practice. The value chain of the provisioning and operation of networks will change and needs to be supported by a technical environment. This environment will need to support a decrease in development time and improved network operations, including further overlapping functions. Opportunity also arises as the role of providing network functionality taken up by a broader range of actors, no longer limited to the vendor of network equipment. This allows a much higher degree of customisation of network functions for the service developer, empowering them with a better deployment and greater control of their service. Conversely, the role of the operator of a network changes as well: instead of just running a system with essentially fixed functionality where only parameterisation is necessary, the dynamics provisioning and deploying new functions become daunting. If, in particular, provisioning will become a frequent action rather than a unique event, it must

be supported by automation that spans across the development and the operational cycle of network functions, as well as orchestration of needed resources—the technical role of an 'orchestration platform' becomes important and has received considerable attention.

Taking such a comprehensive view of development and operation of complex software is indeed a common trend of modern software engineering; it is commonly referred to as 'DevOps' (a contraction of Development and Operations, to highlight the close integration of these two processes). It has not, however, taken root in the actual practice of the telecommunication industry with respect to the operation of networks in an open fashion. This is the core shortcoming that the SONATA project [2] addresses, providing an integrated development and operational process for virtualised network functions, along with the necessary supporting tools like a service-adaptable and resource-adaptable orchestration platforms.

Section 2 provides a perspective of previous results related to SONATA (including research work, standardisation initiatives and common trends in an initial generation pre-production open-source and commercial architectures). Section 3.1 looks into how SONATA's challenges would have to be reflected in a broader 5G architectural context. Specific requirements and challenges arising from various use cases are presented in Section 4. Finally, Section 5 describes the actual SONATA contribution to a 'softwarised' network architecture. Section 6 concludes and previews future activities.

2. RELATED WORK

Network function virtualisation [3] is currently on operators' road maps for deployment, but at the time of writing mostly in a proof-of-concept stage of adoption. However, the arrival of software networks, supported by software-defined networking [4] and NFV technologies, is a universally agreed milestone in the evolution of telecommunications. Several open-source, standardisation and commercial solutions are being developed for NFV management and orchestration [5] ['MANO', as European Telecommunications Standards Institute (ETSI) terms it] in anticipation of its widespread operational use.

2.1. Software development processes and tools for network function virtualisation

There is surprisingly little material available of software development for network functions or resulting services, let alone concrete tools to this end. The Unifying Cloud and Carrier Networks (UNIFY) project [6] delivers some first insights on how to apply the DevOps model to NFV. They provide a multicomponent debugging tool called Epoxide [7]. Compared with these tools, SONATA's approach is a step forward and combines a powerful software development kit (SDK) with a flexible service platform to provide end-to-end support for service developers.

2.2. Orchestration platforms

Simple cloud managers like provide the basic functionality to deploy and manage single predefined virtual machines but cannot handle composed services. Other solutions, like OpenStack Heat [8], Terraform [9] and Application Deployment Toolkit (ADT) [10], are able to deploy entire services but do not focus on network function-specific needs. Projects directly focusing on NFV service orchestration can be divided into three categories. The first consists of research projects, like Network Functions as a Service over Virtualised Infrastructures (T-NOVA) [11] and UNIFY [6]. The second category consists of open-source projects such as OpenMANO [12] or OpenStack Tacker [13] or open software-defined infrastructure [14]. A third category is an initial generation of commercial solutions put forth by telecommunications vendors (expanding their platform, software and integration services to adapt to the disruptive change in the value chain) and supporting technology partners (classically in IT, cloud, etc. but recognising new market opportunities as telecom infrastructure becomes virtualised and software-based). Regardless of their origin, they offer similar functionality and characteristics when confined to the ETSI NFVO specification, and significant differentiation is more apparent when comparing their larger ecosystems that extend beyond the scope of orchestration. This is consistent with the business strategy to provide all NFV-related needs (through their own portfolio or partnerships).

UNIFY's architecture [6] aims at automated and dynamic service creation and recursive resource orchestration. Its global orchestrator includes optimisation algorithms for placement of service components; service-specific actions related to placement and scaling are deployed as a service component. T-NOVA is capable of managing services distributed across several data centres but only provides a simple, rule-based system to control the scaling and placement of deployed services. It also provides a marketplace in which predefined services and virtual network functions (VNFs) can be traded. OpenMANO and OpenStack Tacker aim to be reference implementations of the MANO layer defined in the ETSI NFV Industry Specification Group (ISG) architecture [15], but both are at the beginning of their development. Other orchestration solutions are Cloud4NFV [16] and vConductor [17].

All presented orchestration tools, to a great extent, follow the same principle and try to build a single orchestration solution for different types of services. This creates multiple restrictions for service developers as they cannot influence the orchestration process as such. Some of the existing platforms allow expressing a limited and predefined set of preferences regarding monitoring, scaling and so on. within function descriptions. However, actively influencing service-specific decisions, for example, placement and scaling of services and their components, is not fully supported by any of them. This is possible with SONATA's service platform using function and

DOI: 10.1002/ett

service-specific manager programmes defined by the service developer.

3. SONATA SCOPE AND AN OVERALL 5G ARCHITECTURE

3.1. Scope of SONATA functionality

SONATA aims at increasing the flexibility and programmability of 5G networks with a novel Service Development Kit and a novel modular Service Platform and Service Orchestrator; it will bridge the gap between telecom business needs and operational management systems.

The scope of SONATA is depicted in Figure 1 where the outcome of the project is represented by the Service Development Kit, the Management System and the Service Platform including a customisable Service Orchestrator, a Resource Orchestrator, a Service Information Base along with various enablers; we use here the ETSI reference model as a terminological framework. In fact, while ETSI's division into Service and Resource Orchestrator can be mapped onto SONATA's service platform, SONATA is much more flexible in this regard and allows not only replacing these orchestration aspects individually, but even to change the division of work between these functions if so desired. This is achieved by SONATA's microservices-based architecture (Section 5.1.1); for example, the resource orchestrator corresponds by and large to SONATA's default placement plug-in.

3.2. 5G multi-service management

SONATA advocates a consistent view of 5G network and compute functions, encompassing a wide conceptual range of such functionality. SONATA functionality covers the multi-service control layer and partially integrated management and operation layer and the application and business services layer. SONATA is also capable of incorporating widely heterogeneous physical resources: various access networks (esp., radio), aggregation and core networks, software networks, data centre networks and mobile edge computing clouds.

A multi-service control layer is responsible for the creation, operation and control of multiple dedicated communication network services running on top of a common infrastructure. SONATA's functionality [18] for this layer includes infrastructure abstraction, infrastructure capability discovery, catalogues and repositories, large number of service and resource orchestration functions as plug-ins, information management functionality and enablers for automatic reconfiguration of running services, that is, part of the integrated management layer.

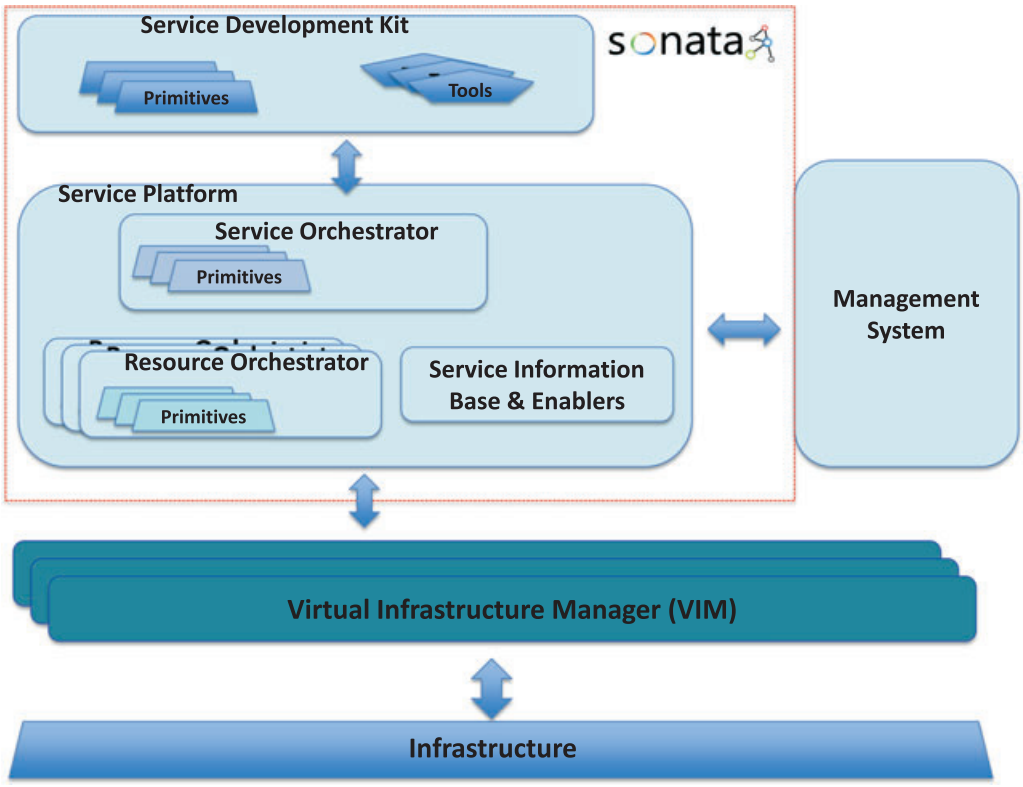

Figure 1. SONATA scope.

Trans. Emerging Tel. Tech. **27**:1206–1215 (2016)

DOI: 10.1002/ett

The business function layer maintains 5G application-related functions, organised in repositories, and DevOps tools necessary for the creation and deployment of services. SONATA's functionality for this layer includes DevOps functionality: catalogues, monitoring data analysis tools, testing tools, packaging tools, editors and basic functionality for application and service programmability.

Figure 2 depicts the way in which SONATA manages various underlying systems. The core idea is to endow SONATA with several infrastructure abstractions, each of which is custom-tailored to the particular needs of the underlying infrastructure. This allows both simple and complex of Virtual Infrastructure Managers (VIMs) to be used, as well as entire networks or single, for example, OpenStack instances to be integrated.

In summary, SONATA's main contribution to 5G networking is efficient integration of service programmability, domain orchestration functionality and DevOps functionality. This will maximise the predictability, efficiency, security and maintainability of development and operational processes around virtualised network functions and chain services.

4. USE-CASE-DRIVEN CHALLENGES AND REQUIREMENTS

4.1. Use cases and resulting requirements

The use cases facilitate the identification of requirements of the SONATA framework while highlighting its full potential for future software networks. SONATA use cases [19] encompass a wide range of Information and Communication Technologies (ICT) domains and their network services as follows:

- Internet of Things: demonstrates SONATA's ability to monitor, classify and optimise Internet of Things network traffic as an enabler of Smart City ecosystem.
- Virtual Content Delivery Networks (CDN): manifests SONATA's capabilities to enhance a virtual CDN service with elasticity and programmability.
- Industrial networks: providing guaranteed, resilient and secure service delivery in vertical industrial networks, such as a wind park.
- Virtual Evolved Packet Core: exhibits and assesses the SONATA's competence to enhance a virtual Evolved Packet Core service in an long-term evolution mobile network.

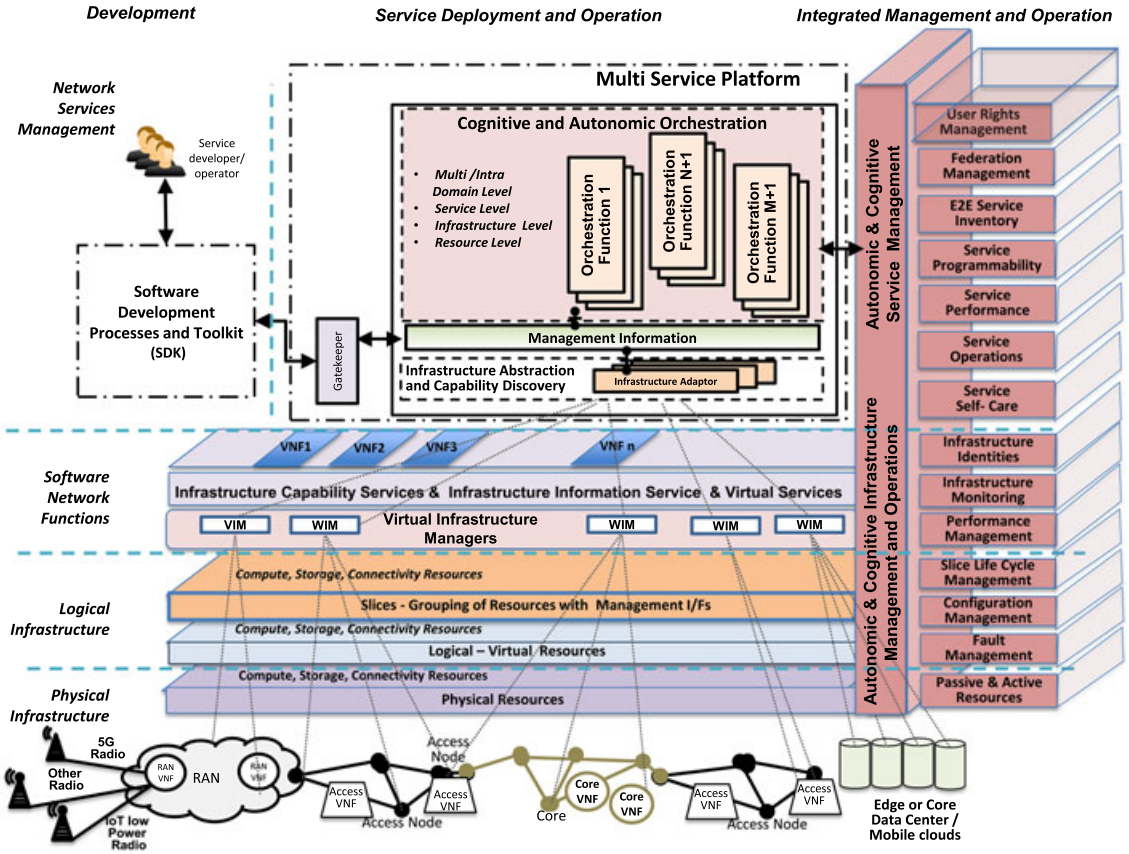

Figure 2. SONATA's relationship to heterogeneous underlying infrastructures.

DOI: 10.1002/ett

- Personal security service: targets the system's potential to offer on-demand network security services to multiple tenants/users with differentiated policies and enable secure network access.
- Separate client and hosting service providers: demonstrates SONATA's support for internetworking between a client service provider and a hosting service provider to accomplish an end-to-end service.

Based on these use cases, we can distil high-level DevOps requirements for both the overall process of developing network functions (SDK) as well as the runtime platform steering the management and life cycle of such functions in an actual infrastructure/network—commonly called the orchestrator.

(1) *Development model requirements*
For the development model, the following requirements are elicited:

- Support for arbitrary network functions: Support for all types of network functions must be possible, irrespective of the technical domain. For example, both core networks as well as wireless fronthaul networks must be supported. To express the different requirements, suitable annotation for network function and services must exist. Scenarios that pertain to both network functions in the narrow sense as well as generic (mobile) edge/distributed cloud computing are in the purview of the SONATA system.
- Network functions form data flow graphs: Individual functions will not suffice to address all needs; it is needed to group functions into general graphs of functions which then constitute the actual service (i.e network service). This is in-line with current NFV specifications (Internet Engineering Task Force (IETF) service chains [20]; ETSI service graphs).
- Reuse by recursive definitions: Reusing such service graphs mandates a recursive notion—a new service can be formed not only by using primitive network functions but also by reusing other, simpler services. Hence, a recursive approach has to prevail in the software development concepts and description techniques.
- Reuse by dynamic composition: Recursive definitions are the first stepping stone. This concept becomes really powerful if existing network functions or service can be dynamically reused and combined into new services without requiring new deployments (assuming the constituting network functions is multi-user-capable).
- Policy-driven support for developer's service: Each service will not only have its own code that deals with the actual data flows but also have additional code that takes care of the deployment and execution process in a manner tailored to the specific requirements of the service—we refer to such code here simply as 'service-specific code' and make this notion more concrete in later sections. For example, scaling or placing a complex service requires decision logic that is rather specific to the particular service; similarly, life cycle management is highly service-specific (e.g. mandating a particular order in which virtual machines comprising a service have to be booted and initialised). This is impossible to achieve in a one-size-fits-all fashion.

As a consequence, we need to support both aspects of a service's code: the actual data-oriented code as well as the caretaking code. This materialises as supporting tools for the developer, in specification for what to ship as a service, and how to be rendered by the orchestration platform, which must be able to execute such service-specific code in a proper context and with proper data.

(2) *Orchestration platform requirements*
Similarly, for the orchestration platform, we can identify the following core requirements:

- No one solution fits all: The large variability of services to be orchestrated, along with their own individual caretaking functions, mandates that the orchestration platform must be able to execute service-specific code, but must ensure proper information hiding and isolation between these code pieces from different services.
- Design to support for evolution and maintainability: Assuming that an orchestration platform will ever have a stable, final form seems overly optimistic. Rather, the platform itself must be able to be extended by additional functionality or to replace existing one. This precludes a monolithic design of the platform itself. A microservices-based approach, with individual modules executing particular aspects (together with the individual functions caretaking code) of an orchestration cycle, seems more promising for adoption by network operators and their bespoke configuration of the platform.
- Support real-world's manifold systems: An orchestration platform should be able to support a wide range of actual infrastructures on which services will be deployed. This ranges from bare-metal over simple hypervisors to full-fledged OpenStack installations conceived of a single logical node. A particular example is sliced infrastructures, where the underlying infrastructure is divided into isolated 'slices', each of which has its own guaranteed resources [21]. In the simplest case, SONATA obtains

Trans. Emerging Tel. Tech. **27**:1206–1215 (2016)

DOI: 10.1002/ett

new slices from an external slicing system (in the extreme case, one slice per service is conceivable). Alternatively, it might be promising to integrate slicing functionality directly into an orchestrator.

- Division and recursion on platform layer: Akin to the idea of recursion to describe services, it is promising to conceive infrastructure and orchestration platform itself in a recursive way, as well. Sub-orchestrators can then be in charge of different domains or different operator networks, working together under the auspice of a higher-level orchestrator. The challenge is to have the orchestrator itself expose an infrastructure-like interface. This recursive approach should also combine nicely with a slicing system.

These high-level requirements and associated challenges are addressed by SONATA—the following sections show how.

5. SOFTWARE-NETWORK TECHNOLOGIES

SONATA project's main goal is to increase the flexibility and programmability of 5G networks in order to bridge the gap between telecom business needs and operational management systems. In this chapter, an overview of the SONATA architecture is provided, including a short description of the main components. The service programming and orchestration framework consist of the SDK, the service platform and different catalogues storing artefacts that can be produced, used and managed by the SONATA system. Services developed and deployed by this system run on top of the underlying infrastructure accessible to the SONATA system via a wide range VIMs; SONATA is fairly agnostic to the particular VIMs.

SONATA's system design is based on the DevOps workflow, which is supported by the integration between the SDK and the service platform. This workflow implies continuous deployment and continuous integration during service development. The main entity exchanged between the SDK and the service platform is the service package to be deployed and runtime information like monitoring data and performance measurements regarding the service package, which is provided to the service developer during the development phase, as well as the runtime. This information can be used for optimising, modifying and debugging the operation and functionality of services.

The main characteristics of the SONATA architecture are explained in Section 5.1, followed by covering the four main actor roles in the SONATA platform (Section 5.2). For a full description of the current state of the SONATA architecture, please refer to the corresponding deliverable [18] at the time of this writing, to be updated during the course of the project.

5.1. Some architectural aspects

(1) A microservices-based orchestration platform

The high-level service deployment procedure is illustrated in the service platform. Each VIM/WAN Infrastructure Manager (WIM) provides the controlling service platform a view of the available resources and capabilities of its underlying infrastructure/network. A gatekeeper module in the service platform is responsible for processing the incoming requests. The service platform receives the service packages implemented and created with the help of SONATA's SDK and is responsible for placing, deploying, provisioning, scaling and managing the services on existing cloud infrastructures. For this purpose, it has modules for orchestrating and managing the complete service chain, as well as managing on the VNF level. All artefacts needed to deploy the service can be fetched from catalogues and repositories. The platform can also provide direct feedback about the deployed services to the SDK, for example, monitoring data about a service or its components. SONATA's service platform is designed with full customisation possibility, providing flexibility and control to both operators and developers. The core mechanism for this is a microservices-based plug-in architecture (Figure 3): all functionality that is to be provided by an orchestrator is assigned to specific plug-ins, all of which are connected to a message bus that ensures correct delivery semantics of all control messages between these plug-ins. Some of these plug-ins implement exactly one function per orchestrator (e.g. the conflict resolver plug-in, which ensures that any possible resource conflicts between services are resolved in a consistent, service-neutral fashion).

Other plug-ins can be customised by the deployed service itself with caretaking code: these plug-ins then act as an executive (akin to Microsoft Window's operating system concept) for this service-specific code.

The service developer can ship the service package to the service platform together with service-specific or function-specific caretaking code, expressing and realising requirements and preferences. Such caretaking code is referred to in SONATA as service-specific managers and function-specific managers, respectively. Service-specific managers and function-specific managers can influence the service and VNF life cycle management operations, for example, by specifying desired placement or scaling behaviour. This grants the developer increased flexibility, control and resilience of their service.

By virtue of a modular design in the Management and Orchestration Framework of the service platform, the service platform operator can customise it,

DOI: 10.1002/ett

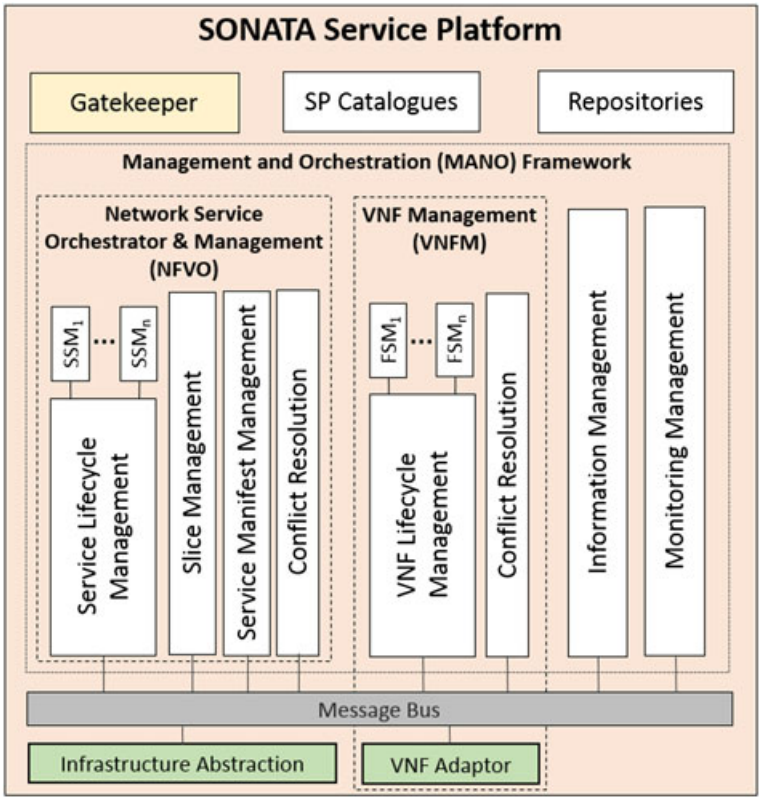

Figure 3. Main plug-ins into SONATA's orchestrator.

for example, by replacing the conflict resolution or information management modules. SONATA's service platform is described in more detail in project deliverable D2.2 [18]. This could be an operator replacing modules of the configurable service platform with alternatives to fit their software network management requirements and preferences.

(2) Recursive orchestration

A recursive structure can be defined as a design, rule or procedure that is (partially) explained using a simplified version of itself. In a network service context, this recursive structure can either be a specific part of a network service or a repeated part of the deployment platform. Although different challenges can be thought of, the general idea of reusing existing patterns could reduce complexity and even add more flexible possibilities for extending the service [22]. In Figure 4, recursive orchestration is shown as a SONATA service platform delegating (part of) the requested service to another instance of a SONATA platform using a dedicated infrastructure adaptor.

Reclusiveness also leads to an easier management of scalability. Monolithic software entities are prone to performance limitations from a certain workload onwards. Scaling by delegating parts of the service to multiple instances of the same software block is a natural way to handle more complex and larger workloads or service graphs. If this reclusiveness is taken into account from the beginning of the development, the advantages of this approach will come at a minimal cost.

(3) An software development kit for network function virtualisation

The SDK supports service developers by providing a service programming model and a development tool-chain. Figure 5 shows an overview of the foreseen SDK components. SONATA's SDK design allows developers to define and test complex services consisting of multiple network functions, with tools that facilitate custom implementations of individual network functions. The implemented artefacts are stored in the developer's private catalogues. Moreover, service components can easily be obtained from external catalogues using the foreseen interfaces. The obtained artefacts can be directly used in a service or after being modified and tested using the SDK development tools. The service components and all the information necessary for deployment and execution of a service are bundled together into a package. The service package can be handed over to the service platform for actual deployment and for testing, debugging and profiling purposes.

Trans. Emerging Tel. Tech. **27**:1206–1215 (2016)

DOI: 10.1002/ett

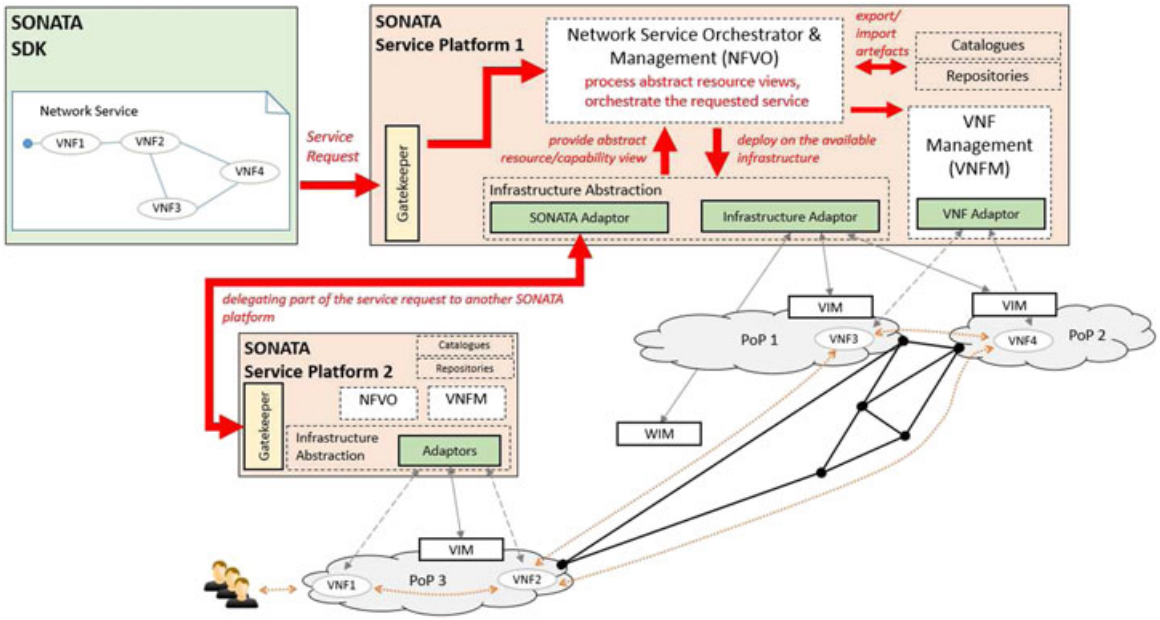

Figure 4. Service deployment using the SONATA framework.

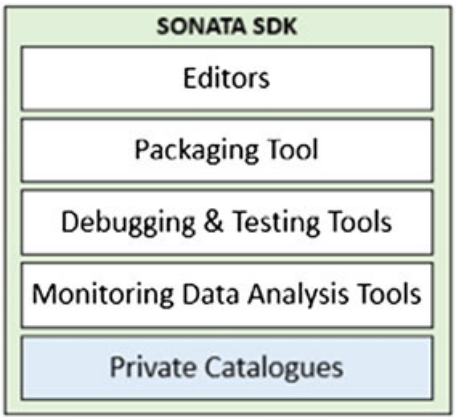

Figure 5. Main components of SONATAs SDK.

5.2. The main actors in SONATA

(1) End users

It is the entity that consumes the network service. Thus, an end user can be a private person or an enterprise, or even a network service provider or content provider. In the SONATA ecosystem, the end user requests the required network services from the network service developer—Figure 6 only shows the simpler case. That is, the end user and the network service developer establish a customer–provider relationship that is regulated by a service level agreement.

(2) Service developer

It is the developer of a software artefact, for example, a VNF or, more specifically, a network service, which can be composed of one or more VNFs. The SONATA SDK shall enable a DevOps environment for the development of network services. The network service developer can use the editor, debugging and packaging tools in the SONATA SDK along with VNF catalogues to compose a network service that can then be moved to the SONATA service platform for deployment and execution. That is, the network service developer interacts with the end user for providing the network services and also interacts with the SONATA service platform operator for the deployment and operation of those network services. However, in the value chain, the developer and provider of such a service can of course be separate entities.

(3) Service platform operator

The service platform operator runs the SONATA platform that manages the execution of network services. The SONATA service platform receives a network service in form of a package that is validated through a series of quality assurance tests prior to their storage in the service platform catalogue. In addition to validation, the service platform operator also manages the deployment and execution of network services on virtual resources made available by an infrastructure operator. As the SONATA service platform supports reclusiveness, an operator can manage multiple SONATA service platform instances, as well. Hence, a responsibility of maintaining smooth operations for a network service, with respect to its corresponding service level agreement, lies on the service platform operator. On one side, the service platform operator interacts with the SONATA

DOI: 10.1002/ett

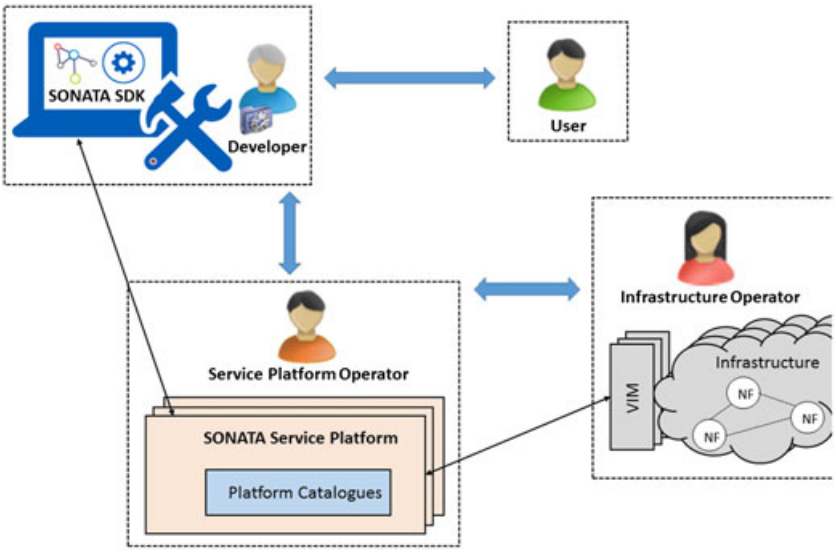

Figure 6. Main components and roles.

SDK for network services reception and, on the other side, interacts with the infrastructure operator for their deployment and execution.

(4) Infrastructure operator

It is the entity that actually operates the physical infrastructure, including computational, communication and storage resources. The SONATA ecosystem does not distinguish between infrastructure owner and infrastructure operator and treat them as the same. This is because the main focus of SONATA is to reduce time-to-market for network services by accelerating and facilitating service development, management and orchestration with DevOps adoption, all while ensuring features such as multi-tenancy, reclusiveness and virtual network slices are supported. The infrastructure operator interacts heavily with the service platform operator as the network services are actually executed in the physical infrastructure. This interaction is enabled by the infrastructure abstraction layer in the SONATA service platform. It is worth mentioning that in most scenarios, the infrastructure operator will be the same as the service platform operator. However, SONATA also supports a use case where the system is engaged by separate entities for these roles, supporting more complex telecom provider business models.

6. CONCLUSIONS AND FUTURE WORK

The SONATA architecture and resulting system push innovation in the evolving state of the art of NFV MANO and network service development. From the approach outlined earlier, we can draw four main conclusions.

Modular and customisable MANO plug-in architecture: the architecture structure provides hitherto unseen levels of flexibility. Third-party developers are empowered with control over specific orchestration and management functionalities pertaining to their own service without being able to influence other services. Similarly, a network operator has a wide range of control over how the service platform should behave. For example, the service platform effortlessly encompasses ETSI's separation into resource and service orchestration, but other orchestration models are supported as well. This boils down into a new notion of **Orchestration-as-a-Service (OaaS)**: the orchestration process itself can be customised on demand by the provider of a service but stays under the operator's control.

Interoperable, agnostic framework: the resulting framework (incorporating both service platform and developer support) is agnostic along multiple axes: it supports multiple VIMs, multiple vendors, deployment at multiple sites. It is also agnostic to whether these functions are used in different kind of networks (mobile, optical, etc.) or in the fronthaul or backhaul part. It is agnostic with respect to whether these services are stateless or stateful.

Efficient network service development and DevOps: SONATA provides service developers with an SDK for efficient creation, deployment and management of network services composed of VNFs along with the required chaining on the platform. The integration of SDK and service platform links an existing gap between service developers and operators.

5G integration: SONATA easily embraces current 5G architecture developments and plays as a good citizen in the entire ecosystem. For example, slicing is well supported in various fashions, interacting with external slicing systems as is bespoke network configuration for industry verticals. Recursion support allows stacked tenant and wholesale deployments in new software networks business models.

Trans. Emerging Tel. Tech. **27**:1206–1215 (2016)

DOI: 10.1002/ett

DevOps prototype: SONATA is an ongoing research project with its first code release scheduled early July 2016. The consortium plans to demonstrate a first prototype showcasing the complete life cycle of a network service in the NFV environment with continuous monitoring by end of year 2016 as well as by exercising several pilot applications that will be chosen among the use cases described in Section 4. The final prototype with full feature set is scheduled for end of year 2017. Using this prototype with its continuously increasing capabilities, we will elaborate on the reduction of development time. Using the SONATA systems, we expect the time to deploy a new network service to be much faster compare with common approaches today. Moreover, we expect a significant reduction of code to write in order to operate network functions and services compared with existing technologies. The proposed DevOps approach raises more technical challenges, for example, related to the update process of running network functions and service. To this end, we will investigate different approaches—again looking at the complete life cycle—and evaluate them in terms of performance, service continuity, resiliency and fault tolerance. Finally, we will compare the SONATA system with existing solutions with respect to resource efficiency and overall service performance.

REFERENCES

1. Chiosi M, Clarke D, Willis P, Reid A, et al. Network functions virtualisation. *Technical Report*, White paper at the SDN and OpenFlow World Congress, ETSI, 2012.
2. *SONATA project*, 2015. Available from: http://sonata-nfv.eu/ [accessed on April 2016].
3. Mijumbi R, Serrat J, Gorricho J, Bouten N, De Turck F, Boutaba R. Network function virtualization: state-of-the-art research challenges. *IEEE Communications Surveys & Tutorials* 2016; **18**(1): 236–262.
4. McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, et al. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review* 2008; **38**(2): 69–74.
5. ETSI. *NFV MANO*. Available from: http://portal.etsi.org/portal/server.pt/community/NFV/367?tbId$=$79 [accessed on April 2016].
6. *Unify project*, 2015. Available from: https://www.fp7-unify.eu/ [accessed on April 2016].
7. Lévai T, Pelle I, Németh F, Gulyás A. EPOXIDE: a modular prototype for SDN troubleshooting. In *ACM Conference on Special Interest Group on Data Communication*, London, 2015.
8. *OpenStack heat*, 2015. Available from: https://wiki.openstack.org/wiki/Heat [accessed on April 2016].
9. *Terraform*, 2015. Available from: https://www.terraform.io [accessed on April 2016].
10. Keller M, Peuster M, Robbert C, Karl H. A topology-aware adaptive deployment framework for elastic applications. In *17th International Conference on Intelligence in Next Generation Networks (ICIN)*, Venice, Italy, 2013.
11. Xilouris G, Trouva E, Lobillo F, Soares JM, Carapinha J, McGrath M, et al. TNOVA: a marketplace for virtualized network functions. In *European Conference on Networks and Communications (EuCNC)*, Bologna, Italy, 2014.
12. *OpenMANO*, 2015. Available from: https://github.com/nfvlabs/openmano [accessed on April 2016].
13. *OpenStack tacker*, 2015. Available from: https://wiki.openstack.org/wiki/Tacker [accessed on April 2016].
14. Mamatas L, Clayman S, Galis A. A service-aware virtualized software-defined infrastructure. *IEEE Communications Magazine* 2015; **53**(4): 166–174.
15. ETSI NFV ISG. *GS NFV-MAN 001 V1.1.1 network function virtualisation (NFV); management and orchestration*, Dec. 2014.
16. Soares J, Dias M, Carapinha J, Parreira B, Sargento S. Cloud4NFV: a platform for virtual network functions. In *IEEE 3rd International Conference on Cloud Networking (CloudNet)*, Luxemburg, 2014.
17. Shen W, Yoshida M, Minato K, Imajuku W. Conductor: an enabler for achieving virtual network integration as a service. *Communications Magazine* 2015; **53** (2): 116–124.
18. *Architecture design: deliverable D2.2—SONATA project*. Available from: http://sonata-nfv.eu/sites/default/files/sonata/public/content-files/pages/SONATAD2.2ArchitectureandDesign.pdf [accessed on April 2016].
19. *Use cases and requirements—deliverable 2.1—SOTATA project*. Available from: http://sonata-nfv.eu/use-cases [accessed on April 2016].
20. *IETF service chaining*. Available from: https://datatracker.ietf.org/wg/sfc/documents/ [accessed on April 2016].
21. Argyropoulos C, Mastorakis S, Giotis K, Androulidakis G, Kalogeras D, Maglaris V. Control-plane slicing methods in multi-tenant software defined networks. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ottawa, 2015; 3–4.
22. Szabo R, Kind M, Westphal FJ, Woesner H, Jocha D, Csaszar A. Elastic network functions: opportunities and challenges. *IEEE Network* 2015; **29**(3): 15–21.

DOI: 10.1002/ett