A Vision for the Next Generation Platform-as-a-Service

Steven Van Rossem[1], Bessem Sayadi[2], Laurent Roullet[2], Angelos Mimidis[3], Michele Paolino[4], Paul Veitch[5], Bela Berde[2], Ignacio Labrador[6], Aurora Ramos[6], Wouter Tavernier[1], Eder Ollora[3] and Jose Soler[3].
[1]Ghent University-imec, [2]Nokia Bell-Labs France, [3]DTU Fotonik, [4]Virtual Open Systems, [5]BT, [6]ATOS

***Abstract*—In an increasingly interconnected world, new opportunities for telecom-based services are emerging. Innovative applications profit from cloud versatility and scalability, but require a platform to combine the optimized 5G network fabric with the advancements in the domain of cloud computing, Software Defined Networking (SDN) and Network Function Virtualization (NFV). In this multi-domain context, we find that available service platforms are lagging, because they tend to be tightly coupled to a constrained set of technologies. In practice, we need the flexibility to deploy different microservices over a heterogeneous range of infrastructure types, aggregating various virtualization, orchestration and control mechanisms. Moreover, the integration of the service requires collaboration among a wide mix of actors (e.g. developers, operators, hardware/software vendors, infrastructure/service providers or vertical integrators). We propose a next-generation Platform-as-a-Service (NGPaaS), devised as a modular framework for the development and operation of network services, while targeting a high degree of both customization and automation. The presented architecture is built around a workflow-based orchestrator which coordinates custom-built tasks across a tailored group of specialized infrastructure or platforms. We also explain how NGPaaS enhances DevOps-principles, to achieve a more efficient integration process across the many isolated administrative domains in the modern telco landscape.**

***Keywords*—NFV; SDN; 5G; PaaS Architecture; DevOps; Dev-for-Operations**

I. Introduction

We consider the Platform-as-a-Service (PaaS) to be a cloud-based environment to support the complete lifecycle of a service, from design to operation and support. By using the PaaS, the service developer or provider can focus directly on the core features of the application. Many auxiliary functions are automated by the PaaS such as service monitoring, scaling, fault mitigation and infrastructure configuration, which alleviate the development and maintenance effort. In a 5G-enabled telecom context however, the notions of 'clouds' and 'cloud-based applications' require an upgraded viewpoint [1]. A traditional cloud application comprehends a client-server setup where the server-side consists of multi-tier setups involving one or more instances of a web-, application- and/or database server which are deployed over one or more datacenters. In a telco-based context, the 'cloud' is expanded with many more operational domains, apps or Virtual Network Functions (VNFs), optimized for pure packet-processing rather than end-user application functionality. New possible target infrastructures to deploy the VNFs are geographically spread across local, edge, access and core networks, allowing e.g. shorter latencies through edge computing, closer to the end-user. From a technical perspective, the drawback is that telco-based services now need to adapt to a wider infrastructure variety (hardware configurations, processor types, operating systems and network/compute configurations, to name just a few), including a large variety of end-point apparatus (from mobile devices over set top boxes to sensors or even self-driving cars). Business-wise, this implies a collaborative ecosystem between many different actors (private/public infrastructure providers, external platform providers, vendors or developers) including challenges regarding security or licensing. An additional goal of the PaaS is to break the silos between the service creation and operation process across the multiple actors with their own operational and administrative domains using an improved telco-grade DevOps approach that we refer to as Dev-for-Operations.

The envisioned PaaS must support a very agile service deployment to fully exploit network softwarization, resource virtualization and network programmability. This requires a wide range of virtualization, orchestration and control mechanisms, across many distributed environments. We therefore split up the required PaaS functionality and implement it following a modular or microservice-based approach. The consequences of this design are intuitively sketched in Fig. 1, where the left side depicts the characteristics of a slow manual service deployment on rigid hardware-based middleboxes, meaning high capital expenditure (capex). To the right side of Fig. 1, fast automated orchestration and configuration of VNFs is illustrated, enabled by NFV and SDN technologies. By decomposing the functionality, a microservice architecture creates better efficiency, productivity, stability, scalability and thus lowers the operational expenses (opex) [2]. The optimal PaaS design is however a trade-off, because too fine-grained flexibility will induce again extra complexity, as well as communication, deployment and testing overhead. High PaaS modularity avoids vendor or technology lock-in, but might have a negative effect on time-to-market, cost or operational performance. To mitigate this, the PaaS must find the right balance between custom-developed service support, which demands a longer development time, and sub-optimal generic operation features which are faster to integrate. The next-generation PaaS (NGPaaS) keeps this in mind by offering a modular framework which allows the plugging in of specialized functionality and the reuse of existing tools.

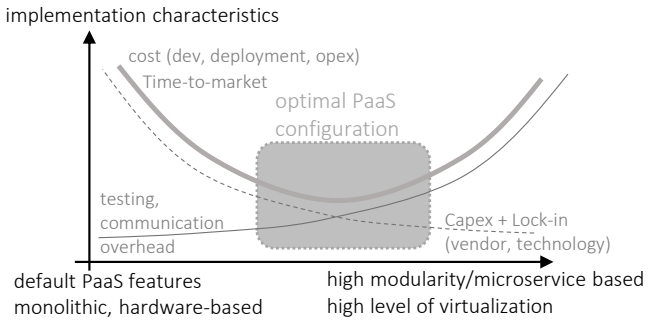

Fig. 1. The PaaS architecture is microservice-based, where the right design achieves the optimal trade-off.

In section II we sketch the challenges to unite different PaaS implementations which each have their own specialized characteristics. Section III analyzes the related work in existing

orchestration frameworks and microservice architectures, while section IV explains in detail our proposed architecture which operates multiple specialized PaaS in parallel. Section V highlights the transformation to a multi-PaaS DevOps methodology which we call Dev-for-Operations. In Section VI, a comparison with the ETSI NFV MANO reference architecture is made [3].

II. Requirements and Challenges for a Build-to-Order PaaS

In the diverse and scattered telecom landscape, there is no one-size-fits-all solution to support the very wide range of 5G, Telco or IoT related services with one single PaaS. We propose a model of multiple specialized PaaS implementations, each targeted at selected virtualization technologies or infrastructure types (e.g. hardware acceleration, edge computing or network control). Moreover, each PaaS can be enhanced with unique features related to telemetry, high availability, autoscaling or SDK toolsets. In Table I, we exemplify three possible PaaS domains with specialized capabilities and supported services.

TABLE I. Example PaaS Types and Supported Services

PaaS type	Telco (fixed access)	5G (mobile access)	IoT
PaaS capabilities	-high resiliency -high security, isolation -datacenter control	-EPC functionality -flexible network control -RAN control	-exploit edge compute -support many access network types, devices and protocols
Service examples	-deploy vCDN -deploy vCPE, firewall	-subscribe to a mobile network -mobile connectivity (voice, data)	-connect a sensor network to a cloud gateway -collect and process sensor metering

Each unique PaaS, like the ones given in the columns of Table I, is generally supporting three main operations to deploy specialized services:

- **Build a service:** Next to the functional implementation, a dedicated runnable component or descriptor is created to exactly define and reproduce the service functionality. A specialized toolset (SDK) can assist. This results in a unique DevOps mechanism per PaaS.
- **Ship a service for deployment:** Using a selected virtualization technique, the created service is packaged and onboarded in the PaaS for deployment. A dedicated repository with versioning control can assist.
- **Run a service:** To deploy and operate the shipped service, the correct configuration and operation workflows must be executed in the associated runtime environment.

Taking the trade-offs of Fig. 1 into account, a compromise is achieved by deploying different PaaS types in parallel and thus enlarging the total set of capabilities. By decomposing existing PaaS implementations we can restructure them using a **build-to-order** principle. A single PaaS is designed once, built out of microservices which can be reused and combined multiple times thereafter. This is illustrated in Fig. 2, where the NGPaaS is in fact a multi-PaaS environment, offering the necessary 'glue' logic for service development and operation across the different actors in this model. In the next sections we dive deeper into the technical implementation of such a modular framework and how it differs from existing platforms.

A. The Multi-Sided Platform

Classic cloud-based service providers tend to have a linear value chain: A fixed and closed set of infrastructure nodes is leased, then specialized software is installed on it and its usage is resold under a different license. The NGPaaS must grow beyond an integrated suite of software products and become the enabler of an open eco-system where the interactions between vendors, developers, service providers and end-customers create added value. Modern cloud-based services are often built around a platform-like business model. In industrial economics, platforms refer to a disruptive organizational phenomenon [4]: the platform organization, which is a 'new type of firm', has a business model that creates value by facilitating exchanges between two or more interdependent groups, usually consumers and producers. One can think about businesses (e.g. Uber, Ebay, Facebook) that offer an online portal where users, cf. sellers or buyers, can interact and generate added value themselves, producing interesting user data to optionally analyze and resell for marketing purposes. With the advent of omnipresent connectivity, these ecosystems enable platforms to scale in ways that traditional businesses cannot. The proposed NGPaaS architecture needs to be designed with this valuable business model in mind. We revisit this in section IV, where the multi-sided platform concept is integrated into our architecture.

B. Specialized Operational Domains

To support a diverse range of PaaS capabilities such as given in Table I, requires integration of a great mix of operational domains, managed by a dedicated PaaS. Various virtualization technologies can be chosen, each with specific characteristics such as virtual machines for extra isolation, containers for enhanced modularity and fast deployment or unikernels for light-weight isolation and quick instantiation. Specialized examples include power-efficient ARM processors or latency sensitive functions running at edge compute nodes or the Radio Access Network (RAN). Existing third-party services can be integrated into a dedicated PaaS. This includes capabilities such as: a public web portal, data storage, big data processing or high availability and load-balancing mechanisms. Specialized PaaS platforms can leverage on FPGA hardware

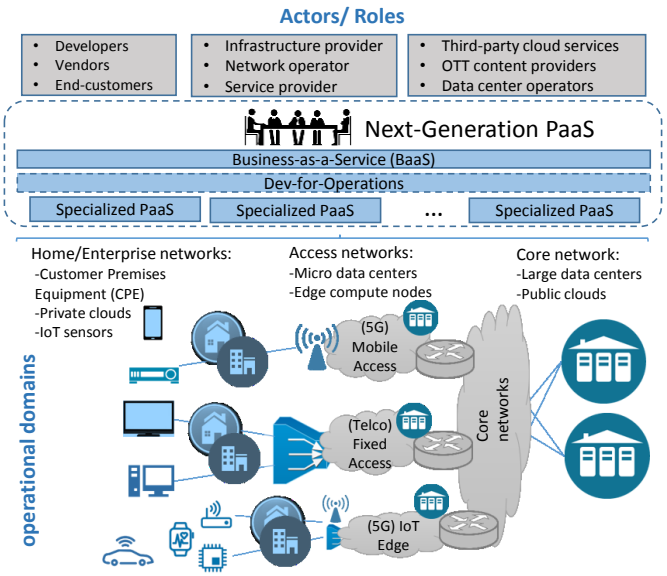

Fig. 2. The next-generation PaaS eco-system unites different actors and combines multiple specialized PaaS domains into a unified operational environment accros many infrastructure types.

accelerators for virtual machines (VMs), containers and unikernels thanks to specific FPGA virtualization technologies [5]. Some examples are introduced in the next paragraphs.

Acceleration can be extensively applied in the Automotive domain. A particular PaaS can integrate the concept of Vehicle-As-Infrastructure, which considers vehicles (such as cars, motorbikes, etc.) as nodes of the infrastructure in which VNFs can be executed. The hardware capability of today's vehicles is continuously growing, to the point where they can execute multiple virtual machines, accelerated through FPGAs or GPUs. In such a scenario, the PaaS can orchestrate functions to be executed in the vehicles, or the vehicles themselves could also be able to migrate workloads to the network infrastructure. FPGA virtualization (together with ARMv8 and Intel support) is very important in this scenario, both to offload and accelerate critical vehicle functionality (e.g. autonomous driving and recognition of pedestrians or signals).

Combining multiple specialized PaaS domains also requires control over the network infrastructure (both physical and virtual). Following the SDN paradigm, SDN controllers (SDNC) can enable a highly dynamic and programmable network control. However, due to the multi-PaaS characteristic of the proposed architecture, a "single controller fits all" approach is not viable. To that end, we envision the ability to build and instantiate a build-to-order SDNC, which meets the requirements of a specific PaaS. Current SDNCs, like ONOS and OpenDayLight (ODL), already offer similar levels of functionality, which the NGPaaS can effectively reuse and extend when necessary. For example the ONOS controller can be packaged and configured with a specific set of features. These features can vary from device specific drivers (e.g. support for different OpenFlow pipelines), to protocol specific drivers (e.g. OpenFlow, P4 and NETCONF) and even high level network applications (e.g. firewalls and mobility support). The different PaaS types presented in Table I (Telco, 5G, IoT), might have very different requirements for their network infrastructure, hence the build-to-order function of a PaaS can take a custom SDNC into account.

In Table II, we list some examples of how to build a PaaS, combining available technologies as described above to implement the earlier explained build-ship-run operations. Each row in this table is a possible PaaS, which can be deployed using the build-to-order principle. The optimal PaaS implementations are chosen (e.g. from Table II) to meet the overall capabilities (e.g. required in Table I). This is a first step towards the practical implementation of the NGPaaS proposed in Fig. 2.

TABLE II. EXAMPLES OF PAAS IMPLEMENTATIONS

Service Type and Descriptor	**Shipping format and repository**	**Execution Environment**
VM creation scripts (Packer, Vagrant) Service descriptor using VMs as VNFs (HEAT, TOSCA, ETSI descriptor, JuJu)	VM images stored in service catalog (JuJu Charms Store, VNF marketplace, OpenStack Glance)	CORD, ONAP, OSM, SONATA Service Platform, KVM, VirtualBox, MANO platforms, OpenStack
Container descriptors (Dockerfile, Docker compose, Kubernetes API object, Mesos Task)	Container image stored in private/public container registry (Docker Hub, Shipyard)	Kubernetes, Mesos, Docker Engine, Docker Swarm, LXC, Rkt, Open Container
Unikernel (OCaml, source code, Click script)	Rumpkernel, MirageOS library, ClickOS images	Modified (Xen) hypervisor, KVM
Accelerators (VHDL, Verilog, DPDK, SR-IOV)	FPGA bitstream, DPDK app, kernel modules	Tweaked x86 platform (hypervisor, scheduler, CPU/Cache/BIOS/kernel optimizations), FPGA platform
SDN network control (custom OpenFlow rules, NETCONF, P4)	SDN app repository (ONOS, ODL applications)	SDNC (ONOS, ODL), virtual/white-box switches
App source code (C/C++, Java)	Compiled app in App Store	ARM / Android / IoT platform

III. RELATED WORK

Many existing platforms attempt to offer a growing set of capabilities. Commercial over-the-top (OTT) platform providers tend to implement their proprietary control and operation mechanisms which only apply to their own data center pool. This complicates the implementation of a hybrid solution where multiple private and public infrastructures must be united. This is tackled by several 5G related research projects**.** They aim to enable multi-domain orchestration across multiple administrations. One solution is imposing a specific model to the infrastructure domains for abstracting available resources to a common orchestrator (5GEx [6]). Another approach is a federated orchestration, where each resource slice has its own orchestrator (5G-NORMA [7]**)**. However, both projects share however the need for an orchestration function which is tightly coupled to the targeted infrastructure domains. A possible solution for including service-specific, customizable logic into the orchestration system is proposed in SONATA [8]. Any service deployed by the SONATA platform can plugin custom-built managers into the orchestration framework. This allows the execution of dedicated placement, configuration or scaling functionalities. But this solution still needs custom adapters for every infrastructure domain or technology. Orchestration protocols or abstraction models need to be defined and standardized before a wide adoption can take place. This limits the flexibility of the platform to quickly integrate third-party operational domains with new capabilities.

Project Superfluidity [9] defines an abstraction model for various flavours of technology components, either generic software (e.g. VM, containers) or specialized software (e.g. unikernels) as well as generic hardware (e.g. x86 servers, white-box switches) or specialized hardware (e.g. GPU, FPGA). This model allows deploying and operating hybrid services in a uniform way, allowing abstraction from their execution environments. This is an important step towards a generic orchestration function.

Several industry-driven consortia such as OSM [10] are building a software stack aligned with ETSI NFV Management and Orchestration (MANO) specifications [3]. This includes multiple dedicated Virtual Infrastructure Managers (VIMs) such as OpenStack, OpenVIM, VMWare or AWS. ONAP [11] is also extending the ETSI NFV MANO architecture with additional telco-grade features. ONAP includes Controllers for various types of infrastructure including network control (using SDN or NETCONF based protocols). ONAP's Policy Subsystem enables the definition of a set of custom rules that

underlie ONAP's control, orchestration, and management functions. This subsystem, as explained in [11], implies however that the total set of supported policy rules and their execution is embedded into ONAP and is not easily modifiable in function of the required PaaS capabilities.

A specialized service runtime environment is the Central Office Re-architected as a Datacenter (CORD) project [12], which provides a virtualized public exchange (central office), architected using datacenter principles. This allows telecommunication providers to run novel connectivity services as VNFs on top of commodity hardware. Using the terminology of the ETSI NFV MANO architecture [3], CORD's main functional blocks are a VNF manager (XOS) and two VIMs (OpenStack and ONOS). CORD is designed using tight coupling between its different functional elements (e.g. between XOS and ONOS), which makes it less flexible to update or plugin new components. But it can be seen as the execution environment for a specialized telco-grade PaaS.

A unique approach to the integration of services on multiple infrastructure domains is done by OPNFV [13]. Through system level integration, deployment and testing, OPNFV creates an ecosystem for NFV solutions. Any participant can bring its own execution environment to the platform (such as ETSI NFV platforms or infrastructure managers). This is used to deploy NFV-based services, brought to the platform by other suppliers. The OPNFV framework links those two parties by installing a Continuous Integration and Development (CI/CD) workflow where the deployment of the services on the execution environments is continually validated across updates. This resembles the workings of an automation server (such as Jenkins), but now expanded to an NFV-based eco-system. Generic execution of workflows, such as implemented in Mistral [14], can mean a valuable enhancement of the orchestration mechanism. Specialized workflows can execute non-default actions related to the service lifecycle, such as a custom update procedure. The added value of using workflow engines in NFV MANO frameworks is also described in [15].

IV. THE MULTI-PAAS PLATFORM

We have introduced the next-generation PaaS as a configurable multi-PaaS environment, tailored to business-defined capabilities. To this end, we foresee a modular framework to easily integrate available third-party services and platforms. To enable a quick time-to-market, we adopt the model proposed by the cloud computing industry consisting of a Business, Platform and Infrastructure layer. This ensures that the whole system remains compatible with modern cloud-based IaaS (Infrastructure-as-a-Service) providers.

A. Roles in an NFV-enabled ecosystem

From a business perspective, we define the role of *NGPaaS Operator* as the actor who defines and controls the overall platform for uniting other actors as depicted in Fig. 2. The *NGPaaS Operator* deploys multiple specialized PaaS at the disposal of *Vertical Service Providers.* The NGPaaS also attracts *Software Vendors* who can supply their own PaaS or service components to the *NGPaaS Operator*. In the operational phase, the *Vertical Service Providers* will request the *NGPaaS Operator* to deploy a set of available services. The *Service End-Users* will affiliate with the *Vertical Service Provider* to get access to the deployed services.

The different architectural blocks and the related roles concerning the NGPaaS are illustrated in Fig. 3. On top sits the Business Layer, where all business-related affiliations take place. After registration, the *NGPaaS Operator* makes the platform functionality available, constrained by regulated access and license management. In the next sections, we will further detail the underlying operational layers.

B. From B/OSS to Business-as-a-Service (BaaS)

In the on top layers, the BSS/OSS model is revisited and re-modeled into BaaS (Business-as-a-Service), a flexible group of available PaaS and services, adapted to the business requirements. Via the interface at ❶ in Fig. 3, IaaS providers can register their available resources and related costs into an **Infrastructure Registry**. From here, a resource pool can be defined that fits into the policy rules (cost-budget and performance requirements) of the business use-case. To promote the usage of its infrastructure, it is in the interest of the IaaS provider to have a dedicated adapter or API available to automate resource provisioning and monitor the status. Dedicated policy rules and alarms processed by the **Conflict resolution**, can steer the **IaaS management** to dynamically scale in/out resources according to the PaaS demand. Available open-source tools which (partly) implement this are e.g. Manage IQ, mist.io or Scalr. The actual NGPaaS operation starts from high-level functionality templates, called Blueprints, which are decomposed into deployable components and mapped to available infrastructure resources. The core functionality can be summarized as a twofold orchestration mechanism (given in Fig. 3-step ❷):

1) **PaaS to IaaS orchestration**: A specialized PaaS (described by a **PaaS Blueprint**) is deployed on an initial set of infrastructure resources. . Once a PaaS is up and running, services can be deployed on it.
2) **Service to PaaS orchestration**: After one or more PaaS are available in the Platform Layer, requested services are orchestrated to a supporting PaaS. The actual deployment and operation of the service is then delegated to the PaaS.

As shown at ❸ in Fig. 3, the *Vertical Service Provider* can request services from the *NGPaaS Operator* to meet its business requirements. In addition, the limits defined in the Service-Level-Agreement (SLA) and the cost-budget are taken into account in the **policy definition**: each deployed PaaS can impose its own policy rules and boundaries in which IaaS resources should be allocated to the services. The **Service Blueprint**, contains all further information to initiate phase two of the above described orchestration mechanism (e.g. VNF or microservice images and scripts).

C. Workflow-driven Orchestration & Operation

The **Blueprint execution** in step ❷ boils down to linking each Blueprint to a set of workflows which do the actual deployment and configuration. A generic implementation is supported by following functions:

- **Blueprint Catalog:** contains the needed workflow descriptors and deployment artifacts for the PaaS and services (e.g. VNF or microservice images, install and configuration scripts).
- **PaaS and Service Records:** contain the dynamically configured parameters, exposed after instantiation of the PaaS or service (e.g. IP addresses, TCP port numbers, resource allocations, access tokens, instance UUIDs).

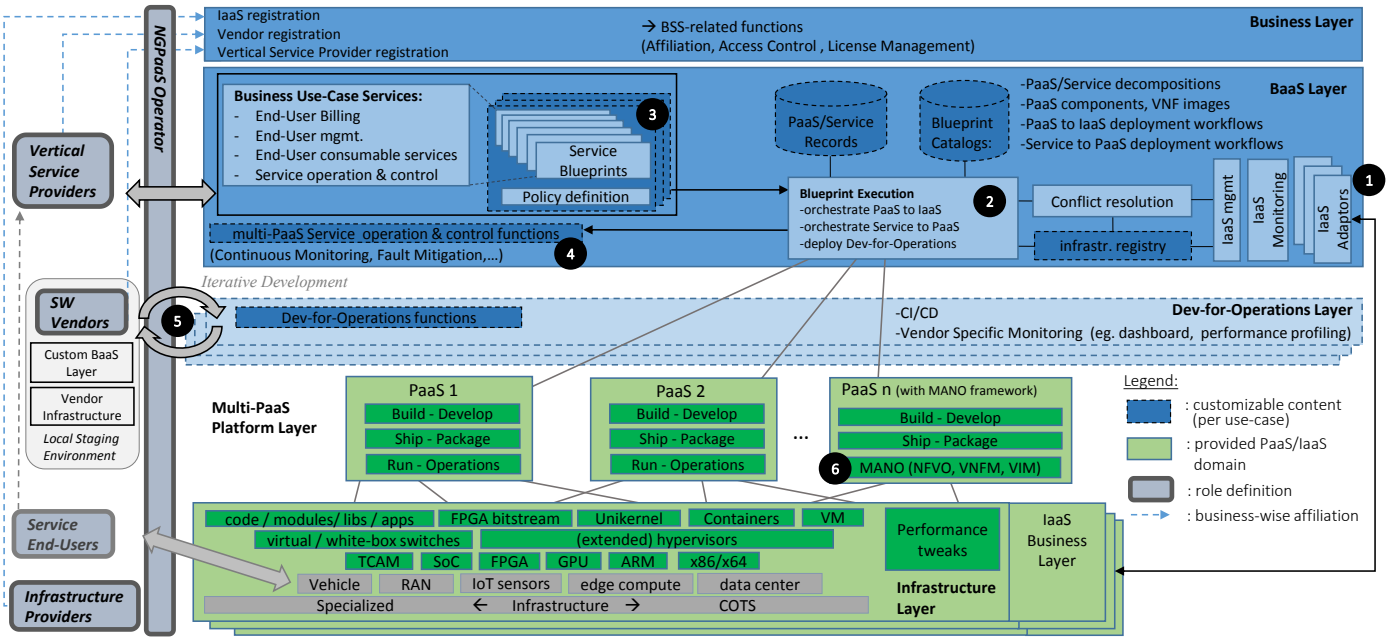

Fig. 3. The architecture of the next-generation Platform-as-a-Service is microservice-based and supports a high degree of customization, automation and an open development interface. In this NFV-enabled eco-system, softwarized services are deployed through selected PaaS domains upon specialized or common-of-the-shelf (COTS) infrastructure.

- **Service Decompose and Mapping Engine:** looks up the available decompositions of the requested PaaS or service so the decomposed functional blocks can be mapped to the supporting IaaS or PaaS.
- **Workflow Execution:** arranges that workflows are orchestrated to the correct IaaS, PaaS and execution engine.

Possible workflow execution engines are given in Table III, which also proposes a model to identify workflow scripts by several attributes. The requested PaaS and services are decomposed into deployable artifacts which can be mapped to the associated workflows by this model. The optional input/output parameters should be processed accordingly.

TABLE III. WORKFLOW MODEL

Input parameters	Attributes	Workflow Execution	Output
-Query from Records -IP, ports -Credentials -Image name -Allocated resources -Template fill (Jinja, …)	-Target PaaS, IaaS -API version -VNF type -Action type (instantiation/ config/update /scale/query)	-Ansible -Chef -Puppet -Mistral -Infrakit -Vagrant, Packer -Terraform -cloud-init -bash scripts	Receive reply after updating: -Records -PaaS/IaaS resources -VNFs Parse the output (regex, translate error to alarm)

Existing service descriptors, e.g. based on the ETSI NFV model used in OSM [17], allow only simple key/value pairs given at deployment time to configure a VNF. More advanced configuration scripts, e.g. for service chaining or scaling, require that configuration information is queried from the available records or is parsed from the output of previously executed workflows.

D. Customized Operation & Control Features

The BaaS layer supports the execution of **multi-PaaS operation and control functions**, as given in Fig. 3-❹. This is a sub-framework where custom-built functions can be plugged in for operational support across PaaS environments. Some operational functionalities cannot be poured into a one-shot executing workflow. Typical features include continuous monitoring, data analysis for fault mitigation, steering VNF updates across PaaS environments for high availability or alarm and log processing for multi-PaaS services. One practical approach can be to run them as containerized processes.

V. FROM "DEVOPS" TO "DEV-FOR-OPERATIONS"

To enable an open interaction between the different actors, our architecture is expanded with a so-called Dev-for-Operations layer. This layer provides platform and service development functionality, closely coupled to the operational PaaS. The DevOps methodology originated in the IT industry to realize a closer and faster collaboration between development and operation teams, within a single organization. The NGPaaS framework aims to extend this 'in-house' feedback flow to a wider, telco-grade context (e.g. multi-vendor, multi-operator) across scattered administrated domains [16]. Adapting to the telco eco-system, the *NGPaaS Operator* needs to collaborate with multiple *Software Vendors*. The main feature of the Dev-for-Operations layer is here to ease the onboarding of new or updated software components in the NGPaaS framework. This process has different stages, as seen in Fig. 3-❺:

- A **custom BaaS Layer** can be deployed in a local staging environment. This way, third-party vendors and developers can design and validate services, closely coupled with the actual operational PaaS environment, because the same orchestration workflows can be tested on a local infrastructure.
- A vendor can upload any new or updated service or PaaS components to the NGPaaS Operator via the Dev-for-Operations layer. A **CI/CD-as-a-Service** mechanism automatically executes unit and integration tests. Only after successful validation, the component can be included in a Blueprint.
- Each external vendor can be granted access to an own, personalized slice or subset of the Dev-for-Operations Layer. This layer can be tailored to each Vendor, with **custom access and execution rights** to monitor, debug or profile a specific service or PaaS component while it is deployed in the operational environment.

Using the described Dev-for-Operations tools, the *NGPaaS Operator* can source components from various vendors and

technologies (avoiding lock-in). The ability to execute many different workflows brings also advantages from a development point of view. A fast DevOps cycle with different development teams is maintained. This way, a large part of the operational accountability can be shared with the Vendors. For example, if a Vendor wants to include its VNF in a service, it must also upload the necessary workflows to deploy, configure and operate the VNF on a supporting PaaS.

VI. Extending the NFV MANO Architecture

The ETSI NFV Management and Orchestration (MANO) has emerged as the de-facto reference for NFV-based platform architectures [3]. We see the NGPaaS as a layer above the current MANO architecture, as we consider the MANO framework as the runtime environment of a specialized PaaS, as illustrated in ❻ on Fig. 3. Many MANO implementations (like OSM, ONAP and SONATA) extend the runtime framework though, with the earlier explained build and ship operations, coming closer to our definition of a PaaS. They include a dedicated design environment, a proprietary service descriptor and catalog. Services are deployed using orchestration (NFVO) and management (VNFM) functions which address virtual infrastructure managers (VIMs) [3]. However, extending a MANO framework with multi-domain orchestration capabilities tends to be an arduous effort [6]. This is because (i) the MANO specific service descriptor and infrastructure abstraction model need to be updated to support new VNF types or operational domains and (ii) specialized VIMs need to be developed and maintained to translate the MANO internal API or message protocol to the APIs of all supported external execution or infrastructure domains.

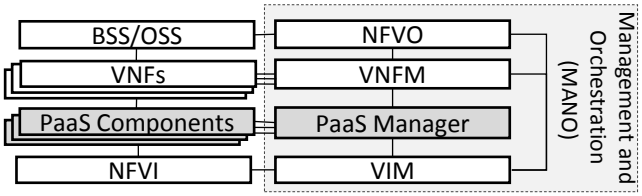

Fig. 4. Updated ETSI MANO architecture, with PaaS capabilities.

In our new NGPaaS architecture, this cumbersome procedure is tackled by shifting to a workflow-based orchestration mechanism. The API or descriptor model of the targeted service or infrastructure manager is addressed natively in the workflow, without an intermediate translation to an internal model. We propose to extend the ETSI MANO architecture as in Fig. 4, with an extra PaaS Layer which extends the VNFM. This supports the twofold orchestration where (i) PaaS components are orchestrated to available infrastructure and (ii) service or VNF deployment and operation can be delegated to isolated PaaS domains. Without the PaaS layer, the ETSI MANO architecture tends to lock itself into an introvert DevOps process, with tightly coupled infrastructure managers (VIMs), orchestrator and service descriptors, where it becomes harder to modify the platform itself. Our proposed NGPaaS architecture on the other hand, promotes a multi-organizational software development model, allowing access to proprietary operational pipelines. The higher-level Business and Dev-for-Operations layers are agnostic about the lower-level PaaS implementation. This facilitates the distribution of operational accountability and integration efforts among different Vendors and Operators.

VII. Conclusion & Future Work

Our proposed platform architecture extends the creation and operation of mobile apps and services, to also the broad range of vertical industries (e.g. automotive systems, smart grid, public safety, health or IoT-based services). The functional platform requirements are defined by the use-case and we enable a tailored approach to realize this platform, with a fast time-to-market. To this end, we create a modular multi-PaaS framework, where specialized PaaS can be plugged in to operate in their own administrative domain, with their own share of infrastructure resources. The Dev-for-Operations layer additionally allows a Vendor-dedicated DevOps cycle, across multiple PaaS domains. This disrupts the current 'cloud platform' paradigm where a fixed combination of options in each layer is imposed, increasing lock-in. The proposed next-generation PaaS is therefore a better suited model with more flexibility and lower barriers to unite the broad spectrum between actors, technologies and services in the modern NFV-based telco landscape.

The architecture of the Next-Generation PaaS, as described in this paper, will be further devised in the European 5G-PPP NGPaaS project. An implementation and demonstration of pilot use-cases are targeted by mid-2019, including VM, container and FPGA based virtualization techniques [18].

Acknowledgment

This work has been performed in the framework of the NGPaaS project, funded by the European Commission under the Horizon 2020 and 5G-PPP Phase2 programmes, under Grant Agreement No. 761 557 (http://ngpaas.eu).

References

[1] van Lingen, F. et al.. "The Unavoidable Convergence of NFV, 5G, and Fog: A Model-Driven Approach to Bridge Cloud and Edge." *IEEE Communications Magazine* 55, no. 8 (2017): 28-35.

[2] Nadareishvili, Irakli et al. Microservice Architecture: Aligning Principles, Practices, and Culture. " O'Reilly Media, Inc.", 2016.

[3] ETSI –NFV, Management and Orchestration. ETSI GS NFV-MAN 001 V1.1.1 , Dec 2014

[4] Hagiu, Andrei, and Julian Wright. "Multi-sided platforms." *International Journal of Industrial Organization* 43 (2015): 162-174.

[5] M. Paolino, et al. "FPGA virtualization with accelerators overcommitment for Network Function Virtualization", RECONFIG '17

[6] A. Sgambelluri, et al. "Orchestration of Network Services Across Multiple Operators: The 5G Exchange Prototype", EuCNC, 2017

[7] M. Rates Crippa, et al "Resource Sharing for a 5G Multi-tenant and Multi-service Architecture", European Wireless Conference, May 2017.

[8] H. Karl et al. "DevOps for Network Function Virtualization: Architectural Approach", ETT Journal , 2016.

[9] Salsano, S. et el. "RDCL 3D, a Model Agnostic Web Framework for the Design of Superfluid NFV Services and Components." *arXiv preprint arXiv:1702.08242* (2017).

[10] OSM - https://osm.etsi.org

[11] ONAP-ECOMP AT&T, "Ecomp (enhanced control orchestration management policy) architecture white paper," 2016. Online available: https://about.att.com/content/dam/snrdocs/ecomp.pdf

[12] CORD http://opencord.org

[13] OPNFV - https://www.opnfv.org/

[14] Mistral - https://docs.openstack.org/mistral/latest/overview.html

[15] Soenen T. et al. "Optimising Microservice-based Reliable NFV Management & Orchestration Architectures", RNDM, IEEE, 2017

[16] Marcus K. Weldon, 'The Future X Network: A Bell Labs Perspective', Chapter 13, March 2016

[17] ETSI, VNF packaging and descriptor, ETSI GS NFV-IFA 011 V2.1.1 Oct - 2016

[18] The Next Generation Platform as a Service (NGPaaS). European 5G-PPP phase 2 project. http://ngpaas.eu/