Agile Service Manager for 5G

Marouen Mechtri[2], Imen Grida Benyahia[1], Djamal Zeghlache[2]
[1]Orange Labs, Paris, France
[2]UMR 5157 CNRS Samovar, Télécom SudParis, Université Paris Saclay, France

Abstract— This paper presents an underlying framework to support and accelerate the production of applications and services in the context of programmable networks (SDN and NFV, clouds). The proposed framework addresses moreover the 5G KPI of "reducing the average service creation time from 90 hours to 90 minutes" as declared by 5G-PPP association in the early of 2015 among other KPIs. The proposed framework relies on SDN, NFV and Cloud principles and technologies and proposes extensions towards the end to end abstraction that is required for automation of service production. A Service Manager Architecture fulfilling the agility, acceleration and automation requirements is presented along with its relationships and interfaces with the applications and network levels. An application requiring network services, expressed in a network service descriptor, is used to illustrate the architecture usage and benefits and highlights the remaining future research needs and trails.

Keywords— Service Management, 5G, SDN, SFC, NFV, Cloud

I. INTRODUCTION

The notion of abstraction is essential for agile services creation so that programmers consume an underlying service without knowing much about it other than its inputs, outputs and functionality. Given such abstraction, there are two different on-going creations. The first is the creation of the services themselves by assembling lower level services. The second is the generation of these lower level services, or abstractions. There are consequently two roles involved, one that architects the service and the second that architects the low level services and associated resources. Agile services are created by first identifying and building the lower-level abstractions upward from the resources. Once a catalog or inventory of these underlying services is defined and made available, service architects can use and assemble them into functional systems known as ''services''. What is very important to make a service agile is to minimize, ideally eliminate, the need to reflect or propagate up to the service definition changes or actions on the low level services. These changes should not affect or modify the service order templates and workflows. More precisely, the low level services changes have to be hidden from the service orchestration process. In the context of SDN, NFV and Cloud, this means that changes in real devices or virtual infrastructures (containers, networks and networking functions) have to be hidden from the service definition and workflows. Consequently, the provisioning of different abstractions and their deployment on different virtual platforms can vary (or can be altered) without changing the service and the way it is assembled or produced. This is the motivation and objective of this paper that focuses on building the SDN, NFV and Cloud abstractions to ease and accelerate higher level services or applications production. The proposed framework addresses then the 5G-PPP [1] (Public-Private-Partnership) key performance improvement indicator (or objective) of *"reducing the average service creation time from 90 hours to 90 minutes* to ensure an efficient path towards 5G networks and services.

To be more specific, this paper focuses on the combination of the Cloud, SDN and NFV realms and their interactions with service orchestration to build the needed abstraction. This is achieved by conducting an analysis of available open source, commercial and academic solutions, tools and software in order to identify gaps in the candidates and propose extensions in service descriptions and interfaces. The degree of abstraction is also taken into account since both virtual and physical resources are expected in the low level services. Following current trends in clouds and software networks, the service level (service and application developers, programmers and providers) would probably like to be made aware of this but definitely does not want to define the services all the way from top to bottom (at all development levels). The functional split between the three domains, the orchestrator and the applications is consequently also considered in our work that proposes a service manager and the underlying framework to ensure faster production of services and make the services agile.

II. BACKGROUND ANALYSIS AND RELATED WORK

In this section, we analyze existing lifecycle management architectures in Cloud, SDN and NFV and their associated languages, interfaces and service description templates. These architectures were developed in parallel and rather separately in the past. They still need to be bridged into a more cohesive service creation and management framework to enable applications and services to take full advantage of future softwarized network architectures and virtualized network functions.

Our objective is to analyze the current state of art on service lifecycle management solutions, tools and methods to assess their usability for agile and faster service production and deployment far beyond what can be achieved today. This is also a clearly stated goal in various roadmaps and especially that of 5G-PPP whose aim is to reduce the production of new applications and services from days (90 days) to time scales of the order of the hour (with a long term ambition to bring this time down to 90 minutes). Achieving this type of objective, of producing and deploying and putting in exploitation a service rapidly, requires relying on key advances in networks and

978-1-5090-0223-8/16/$31.00 © 2016 IEEE

networking services. The current push for network abstraction layers and networking APIs in SDN and Network Functions Virtualisation enable the programmability and control of network services and functions by management layers as well as applications or higher level complex services.
The question is hence to identify the systems that can form an efficient and powerful modeling, control and management set to speed up automated service production. Starting from a user expressed need to the deployment and activation of the required service followed by the management of its life cycle.

There is currently no comprehensive and cohesive unique system that can fulfill the requirements for agile service production and management. Cloud services production and management solutions and frameworks are gradually and incrementally filling some of the gaps and are a good starting point to reach the stated objective. Other service life cycle management frameworks, such as ITILv3 [3] (Information Technology Infrastructure Library), that describe the life of an IT service (from planning and design to delivery and continuous improvement) are also used for IT lifecycle management. They are known as Infrastructure Management Services and are defined as a set of concepts and rules for managing infrastructure, development and operations within information technologies but have not been designed and thought for cloud services from the onset. TOSCA (Topology and Orchestration Specification for Cloud Applications), from OASIS [10] [11] (the Organization for the Advancement of Structured Information Standards), on the other hand, aims at enhancing the portability of cloud applications and services. TOSCA is meant to enable interoperable description of application and infrastructure cloud services independent of the suppliers creating the service, the cloud providers or the hosting technologies. Since we expect to combine principles from the Cloud, SDN and NFV to move towards agile service creation, production and management, we foresee cloud services lifecycle management solutions and architectures as natural candidates. We start consequently by describing cloud services lifecycle management. Continue with service management in SDN and NFV to derive a comprehensive framework for agile services.

1) *Cloud services lifecycle management*

The objective of cloud service lifecycle management is to manage the dynamic nature of the cloud, automate the provisioning, facilitate flexibility and meet the users' requirements. In [4], authors propose an integrated methodology that covers the entire lifecycle of virtualized services on a cloud environment and propose metrics to track in each phase of the lifecycle. They divided the cloud service lifecycle into five phases:

- ***Service requirements:*** The consumer details the technical and functional specifications that a service needs to fulfill and also specifies non-functional attributes (e.g. constraints and preferences on data quality, service compliance and required security policies for the service).
- ***Service discovery:*** searches (or discovers) in the cloud service providers that offer services matching the specifications detailed in the requirement stage.
- ***Service negotiation:*** covers the discussion and agreement that the service provider and consumer have regarding the service delivered and its acceptance criteria. Service acceptance is usually guided by the Service Level Agreements (SLA) that the service provider and consumer agree upon.
- ***Service composition, orchestration:*** In this phase one or more services provided by one or more providers are combined and delivered as a single service. Service orchestration determines the sequence of the service components.
- ***Service consumption and monitoring:*** The service is delivered to the consumer that will require tools that enable quality monitoring and service termination if needed. This phase spans both the consumer and cloud areas as performance monitoring is a joint responsibility.

The cloud service lifecycle was also defined in [5] and [6] as the following succession of operations:

- ***Design-time operations:*** cover service description and semantic information; general information that allows the service to be published.
- ***Provisioning operations:*** cover aspects of describing offers and service publishing, discovery and contracting.
- ***Deployment operations:*** cover service instantiation and commissioning.
- ***Execution operations:*** cover managing and billing aspects.
- ***Retirement operations:*** cover service retirement.

Other lifecycle management solutions follow typically the same phases and steps but all tend to use their own specific terminology to describe the phases. A commercial solution, BMC Cloud Lifecycle Management solution [7], supports the full lifecycle from request to retirement. This solution allows the organizations to deliver flexible and customizable cloud services while maintaining a structured, controlled, and dynamic IT environment. Cloudify [15] is another cloud service provisioning , control and management framework that has the interesting feature of relying partially on the TOSCA framework and interacting and interworking with a very popular cloud stack, namely OpenStack [2]. Evidently, in order to provide clouds services production and management solutions, we will favor open service management components and systems and even more so open source components such as Heat [14] even if it only addresses orchestration via templates (named HOT) and hence is limited in scope when referring to our objective of proposing a service manager capable of interacting with cloud, SDN and NFV architectures. The cited solutions do not fulfill all the requirements and will have to be extended and combined with SDN and NFV services, interfaces, components and their descriptors to provide viable solutions for agile services.

2) *Service management in SDN and NFV*

Service management in SDN and NFV is currently focused on the concept of orchestration in support of service lifecycle management and is just one component of the overall service management.

In the recommendations of the ETSI-NFV proposed in [18], the MANO architecture has an NFV Orchestrator which is responsible for:

- NS lifecycle management (including instantiation, scale-out/in, performance measurements, event correlation, termination)
- global resource management, validation and authorization of NFVI resource requests
- policy management for NS instances

NetCracker's Service Orchestrator [19] enables end-to-end orchestration of service provisioning for hybrid networks made of both virtualized SDN/NFV-based components and traditional network technologies. No matter the specification and implementation available in the literature the merging of cloud, SDN and NFV in a cohesive system still requires further research and development.

3) *Abstractions related to Service management*

The abstractions of interest in this work are service descriptions and interfaces that can speed up combined cloud and network services production and make the high level service agile. Current service descriptions and service templates used to express a service request that are likely to fulfill the agile service goal and can cover cloud, SDN and NFV are YAML [8], JSON and XML. YAML is the most expressive and more natural language for describing high level complex services when compared to XML and JSON. These service description languages are used to make complex services queries to the service lifecycle management architecture as depicted in Figure 1. The service lifecycle management interacts with the cloud, SDN and NFV infrastructures resources and low level services to ensure service provisioning, control, monitoring and management. The interactions between the service life cycle management and the underlying infrastructures also require the specification of a second level of interfaces and APIs that can address both traditional cloud services and advanced networking services and network functions.

In the current state of the art, the most appropriate and most expressive and extensible cloud service specification seems to be TOSCA that uses a service description framework (with service templates, languages and grammar) for cloud services supported by industry leaders such as Cisco, IBM, RED HAT, etc… TOSCA defines key service templates and components for service description and service life cycle management that can be summarized as follows:

- ***Topology Template:*** that defines the structure of a service and consists of a set of Node templates that model the components of the workload and Relationship templates that model the relations between the components.
- ***Node Types and Relationship Types:*** It describes respectively the possible kinds of components and the possible relationships between them. These types allow defining lifecycle operations to implement the behavior an orchestration engine can invoke when instantiating a service template (e.g. a node type can provide lifecycle operations: 'create instance', 'start', 'stop'...).
- ***Plans:*** defined in a Service Template describe the management aspects of service instances, especially their creation and termination. These plans are defined as process models i.e. a workflow of one or more steps. Plans in TOSCA use existing workflow languages such as Business Process Model and Notation [BPMN 2.0] or the Business Process Execution Language [BPEL 2.0] and can also use any workflow language supported by a management environment. TOSCA recommends BPMN 2.0 as workflow language to model management plans as it offers a standardized graphical rendering and does not force the workflow graph to be acyclic [12] Plans are modeled by the developer of the application or experienced operators ensuring widespread usage of their accumulated best practice knowledge [13] Plans orchestrate the different management operations offered by the nodes to fulfill their task. The TOSCA specification defines three types of management plans: Build, modification, and termination plans.

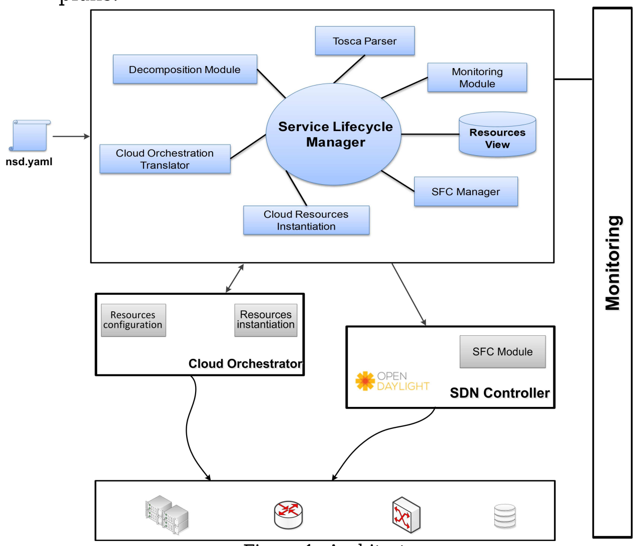


Figure 1: Architecture

Currently, the state of the art operates in each of the three domains (Cloud, SDN and NFV) independently with specific architectures for each domain. For instance TOSCA addresses well cloud services and resources and their use to compose complex cloud services but can not describe or use network services since they are not part of its specification and there are no descriptors for networking services and no means to include virtualized network functions and their connectivity and their chaining. There is a need to bridge and combine these domains by proposing appropriate extensions to build a comprehensive (all inclusive) service request description that can encompass cloud, network and control services.

III. Proposal : Service Manager Architecture

The problem of Cloud resource orchestration has been increasingly studied in recent years and led to the emergence of new cloud orchestration tools. Existing cloud resource orchestration frameworks like Cloudify and OpenStack Heat are mature enough for traditional cloud services (compute, memory, storage) but are unable to provide service function chaining [16] [17] (SFC) across networked cloud infrastructures. Service function chaining is becoming very popular and considered as one of the most important services a provider has to offer. Our proposed architecture aims at enabling service function chaining over cloud and SDN environments. The idea is to take advantage of existing Cloud orchestration and SDN management tools and enhance the architecture with 'missing pieces' and a service lifecycle manager (SLC) to support complex applications while enabling SFC features. The architecture described in Figure 1 meets the need for coordination and orchestration with managers and orchestrators of cloud resources and SDN controllers. The proposed SLC manager coordinates with the cloud orchestrator to set up virtual resources acting as containers for user applications and virtualized network functions. The SLC manager interacts with the SDN controller and their applications to provide user application with the control and the programmability of the network part by setting up the required virtual network functions and the associated network forwarding graphs.

The SLC manager handles service requests (described with “yaml” in our case) from applications. The SLC Manager parses the request and calls its internal modules to process the demands and provides users with agile service deployments. The key modules of the SLC manager are:

- **Tosca Parser:** responsible of parsing requests. In our implementation, the parser is an extended version of the basic Tosca parser that considers just Cloud resources. We enhanced the Tosca parser and its associated grammar so it can deal with both cloud and network resources.
- **Decomposition Module:** After the request is parsed, this module divides the resources into two types: resources/services to be deployed by the Cloud Orchestrator and resources/chaining paths to be deployed by the SDN manager.
- **Cloud Orchestration Translator:** translates the requested Cloud resources to the template language of the Cloud Orchestrator (Cloudify or Heat in our implemented solution).
- **Cloud Resources Instantiation:** after translation of the request, the Cloud Resources Instantiation module communicates with the Cloud Orchestrator to instantiate the resources.
- **SFC Manager:** This module deals with networking and chaining demands. After the resources are divided by the Decomposition Module, the SFC Manager calls the SDN controller to instantiate the resources.

Note that the decomposition module handles the terminology mapping between the IETF and ETSI-NFV standards. This mapping is at this stage necessary because for example the Tosca parser uses the ETSI-NFV terminology whereas the SFC module of OpenDaylight relies on the IETF terminology. The terminology mapping is depicted in Table 1.

ETSI-NFV		IETF	
Virtualized network function	VNF	Service function	SF
Connection point	CP	Service Function Forwarder	SFF
Virtual link	VL	implicitly	-
VNF forwarding graph	VNFFG	Service Function Chain	SFC
Network forwarding path	NFP	Service Function Path	SFP

Table 1: ETSI-NFV to/from IETF SFC terminology mapping

IV. Zoom on Key Modules

In this section, we will describe the main components of the proposed service lifecycle manager. These components ensure the management of the service starting from its description at request time to the instantiation and the deployment steps of the entire service. When the components are extended with additional features to include network services, the extension is visible in the description.

A. Extension of the TOSCA template

The TOSCA language proposes a grammar for describing service templates for cloud applications. These templates are essentially composed of:

- ***Topology template:*** which describes the cloud application components using Node, Relationship and Groups elements,
- ***Plan***: that describes actions and sequences that enable the deployment of the topology template,
- ***Node types, relationship types and artifact types***: are defined for reuse needs e.g. a database can be defined as a node type and can be instantiated several times in the topology template as node template.

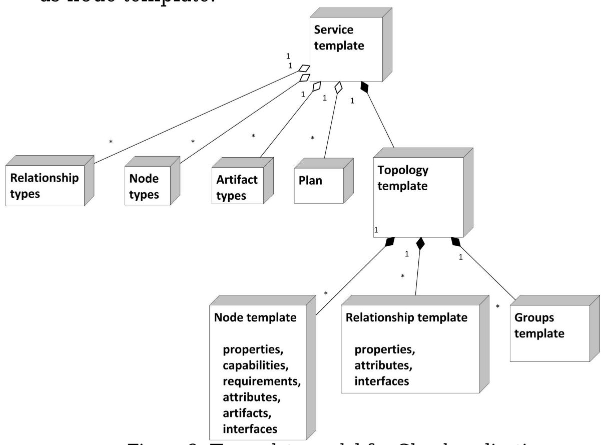


Figure 2: Tosca data model for Cloud application

Figure 2 represents the basic and essential elements and attributes of the TOSCA grammar that are used for abstract representation of cloud application components.

The topology template can be seen as a directed graph with Node Templates as vertices and Relationship Templates as edges connecting them. The Group Templates form sub-graphs of the Topology Template graph.

The node template represents an abstraction of the application or the resource requested by the user. Let us take an example of a mysql database connected to a webserver with these two nodes hosted on a VM. These three nodes (webserver, mysql and VM) can be represented as a node template and the relation between them (hosted on, connected to) as relationship template. The abstraction offered by TOSCA it is generic enough to support the representation of VNFs and their stitching to from a VNF service chain (VNF-FG in the ETSI terminology). The TOSCA data model as depicted in Figure 2 is however specialized and limited to cloud applications. In order to embed network services, this model has to be extended.

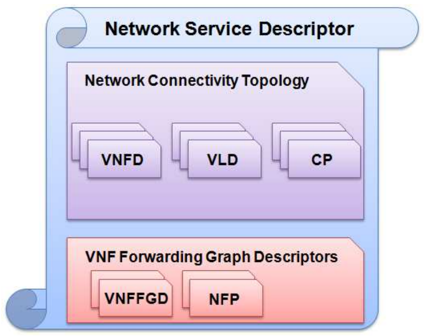

Network Service Descriptor
Network Connectivity Topology
VNFD
VLD
CP
VNF Forwarding Graph Descriptors
VNFFGD
NFP

Figure 3: Network service descriptor with the extended Tosca

Based on the ETSI-NFV standard and terminology, there are two types of graphs composing the network service descriptor [17] and these must be included in the TOSCA model. The extended model is depicted in Figure 3 that reveals two types of graphs to enrich the data model of TOSCA to support NFV:

1. The first one is the network connectivity topology (NCT) that specifies the virtual network function (VNF) nodes that compose the global service and the connection between them using virtual link (VL). Each VL is connected to VNF through the connection point (CP) which represents the VNF interface.
2. The second type of graph is the VNF Forwarding Graph (VNFFG) and is established on top of the Network Connectivity Topology. The VNFFG is composed of network forwarding paths (NFP) that are ordered lists of Connection Points that form a VNF chain.

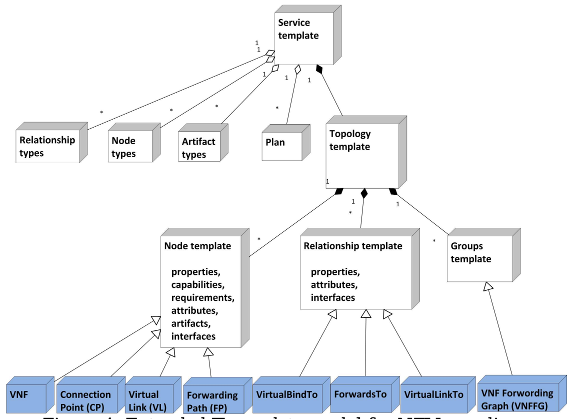


Figure 4: Extended Tosca data model for NFV paradigm

The proposed extension (see Figure 4) of the TOSCA grammar and the associated parser use the notion of node template by inheriting new nodes to represent the VNF, CP, VL and NFP components. The VNF node is characterized by the associated VM and the network function appliance that will be hosted on the VM. Note that the network function appliance will be represented as an artifact element in the grammar of TOSCA. The VL represents the virtual link that interconnects the VNFs. To establish this connection, we use the new inherited relationship nodes to bind the CP to the VNF (using the "VirtualBindTo" relationship) and to link the CP to the VL (using the "VirtualLinkTo" relationship). The forwarding path (FP), inherited from the node template, is modeled as an ordered list of CPs forming a sequence/chain of VNFs that will be traversed by packets or traffic flows. The FP is connected to CPs using the relationship "ForwardsTo". Note that each VNF forwarding graph includes a list of FPs that are interdependent and have common characteristics like having the same policy rules (e.g. several FPs have the same source and destination IP addresses). For this reason, the VNFFG is modeled as a Group and thus inherits from Groups template.

B. SFC Manager

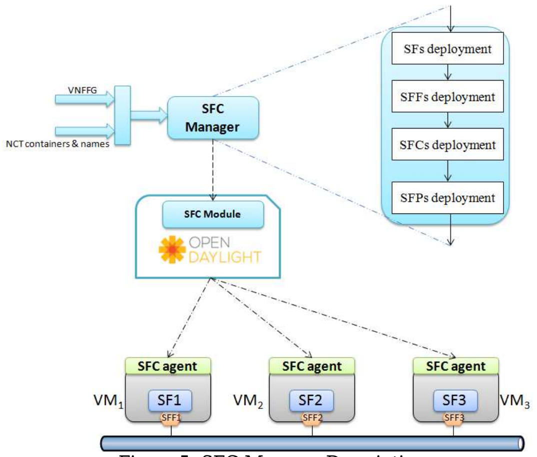


Figure 5: SFC Manager Description

After deploying the required VMs and the associated networks using the cloud orchestrator, the SFC Manager interacts with the SDN controller to deploy and instantiate the VNFs and their associated VNF Forwarding Graphs. The SDN controller is invoked to achieve an SDN oriented stitching of the VNFs (connecting the SFs more precisely); this is especially relevant when classifying flows with respect to their associated chains or NPs. The SDN controller used in our architecture, selected from the open source development communities, is OpenDaylight since it proposes in its Lithium release a module to deploy the VNFFG. This ODL SFC module was nevertheless extended because it does not fulfill our requirements and is not completely mature for the purpose.
Note that the decomposition module translates the terminology of TOSCA which is based on the ETSI-NFV standard to the terminology of IETF standard since OpenDaylight is based on the IETF terminology. As depicted in Figure 5, the proposed SFC manager interacts with the SFC module running on the OpenDaylight to deploy the SFCs.
The steps executed by the SFC Manager are:

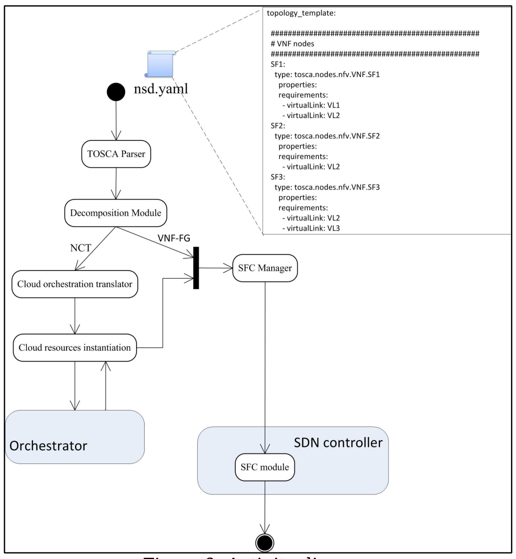
- deployment of SFs (VNFs);
- deployment of SFFs (CPs);
- deployment of SFCs (VNF-FGs);
- deployment of SFPs (FPs).

These configurations are then sent by ODL to the VMs (already deployed by the Cloud orchestrator) through the SFC agent (installed in the VMs).

V. SLC MANAGER OPERATION

Figure 6 describes the different interactions between the SLC Manager components when deploying a network service (through the nsd.yaml template). The instantiation of this template requires a communication between the SLC

Manager, the Cloud orchestrator and the SDN controller since we also have to deploy VNFs and their chains in this complex service use case. The first step performed by the SLC Manager is the validation of the nsd.yaml template. Then, the decomposition module generates two new graphs. The first graph (NCT) describes the required VMs capacity, their connectivity and the software they are required to host (e.g. SFC agent). The second graph generated by the decomposition module is the VNF-FG. This graph defines the requested chains and their forwarding paths. This graph is only deployed once the VMs are successfully instantiated by the orchestrator and provided with addresses (by the "Cloud resources instantiation" module). The third step done by the SLC Manager is the translation of the NCT graph to the language used by the Cloud orchestrator (e.g. cloudify blueprint, heat orchestration template…). The fourth SLC Manager step instantiates the cloud resources by sending the created template in the previous step to the cloud orchestrator. Finally, the SFC Manager deploys the VNF-FG when receiving the order from the "Cloud resources instantiation" module and the VNF containers/VMs addresses.



```
topology_template:

  ##################################################
  # VNF nodes
  ##################################################
  SF1:
    type: tosca.nodes.nfv.VNF.SF1
      properties:
      requirements:
        - virtualLink: VL1
        - virtualLink: VL2
  SF2:
    type: tosca.nodes.nfv.VNF.SF2
      properties:
      requirements:
        - virtualLink: VL2
  SF3:
    type: tosca.nodes.nfv.VNF.SF3
      properties:
      requirements:
        - virtualLink: VL2
        - virtualLink: VL3
```

Figure 6: Activity diagram

VI. RESEARCH ORIENTATIONS AND CONCLUSIONS

The coming 5G networks are characterised by new network technologies (e.g. new technologies for the Radio Access Network) but also by the massive connectivity of heterogeneous connected devices and the transformation of network and services into a set of software modules assembled in agile and flexible manner to fulfil a given customer demand and context. In this paper we addressed a 5G KPI that is of "reducing the average service creation time from 90 hours to 90 minutes". The study of this KPI leads us to review the different phases of service creation while focusing on the service lifecycle management phases. The state of the art reveals that no comprehensive and cohesive framework for service management in the softwarized environment is available: during its different phases, different skills, stakeholders are involved. This is intensified by the use of different tools, parameters and configurations with different various level of details. Towards agility required by 5G, abstractions are cornerstone: Abstraction includes the languages, templates and grammar to describe the networking service needs in terms of VNFs, VLs and CP for example. It also includes the software modules that hide the network complexity while relying on inner service components and resources: Service Manager, and resource orchestrators etc. In this paper, we developed software modules for an agile service manager with its relationships and interfaces with the applications and network levels, while relying on open source platforms and extending well known templates to ensure agility. The next steps would be to focus more on the definition of abstractions in terms of creating new extensible DSLs covering SDN, NFV and clouds while taking into account the different initiatives from standards bodies like ETSI NFV, ONF CIM (Common Information Model) and TMF Zoom IM. Towards agility we will also focus on specifying and developing the needed mechanisms to ensure fluid and dynamic usage of different DSLs for the different phases of service lifecycle management.

VII. REFERENCES

[1] http://5g-ppp.eu/

[2] OpenStack project, http://www.openstack.org/

[3] OGC-V3-Intro: "The Official Introduction to ITIL Service Lifecycle" (TSO, 2007, Fifth edn. 2007)

[4] Karuna P Joshi, Tim Finin and Yelena Yesha, "Integrated Lifecycle of IT Services in a Cloud Environment", University of Maryland, Baltimore County, Baltimore, USA

[5] V. I. Munteanu, T.-F. Fortis, and V. Negru, "Service lifecycle in the cloud environment," in Symbolic and Numeric Algorithms for Scientific Computing, 2012. SYNASC. International Symposium on, September 2012.

[6] T.-F. Fortis, V. I. Munteanu, and V. Negru, "Datastores supporting services lifecycle in the framework of cloud governance," Scalable Computing: Practice and Experience, vol. 13, no. 3, 2012.

[7] BMC 'Cloud Lifecycle Management'. http://www.bmcsoftware.fr/it-solutions/cloud-lifecycle-management.html

[8] http://yaml.org/spec/current

[9] http://www.yaml.org/spec/1.2/spec.html#id2708649

[10] http://www.cloud-council.org/CSCC-Webinar-OASIS-TOSCA-Enabling-Application-Portability-in-the-Cloud-10-22-14.pdf

[11] http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.pdf

[12] Kopp, O., Martin, D., Wutke, D., Leymann, F.: The Difference Between Graph-Based and Block-Structured Business Process Modelling Languages. Enterprise Modelling and Information Systems 4 (1) (2009) 3–13

[13] Kopp, O., Binz, T., Breitenbücher, U., & Leymann, F. (2012). BPMN4TOSCA: A domain-specific language to model management plans for composite applications. In J. Mendling, & M. Weidlich (Eds.), Business process model and notation (Vol. 125 of Lecture Notes in Business Information Processing, pp. 38—52).

[14] Heat 'OpenStack Orchestration service', https://wiki.openstack.org/wiki/Heat

[15] Cloudify project. http://getcloudify.org/

[16] Paul Quinn, Jim Guichard, "Service Function Chaining: Creating a Service Plane via Network Service Headers", Computer, vol.47, no. 11, pp. 38-44, Nov. 2014, doi:10.1109/MC.2014.328

[17] SFC header: https://datatracker.ietf.org/doc/draft-zhang-sfc-sch/?include_text=1

[18] ETSI GS NFV-MAN 001 http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf

[19] NetCracker Service Orchestrator, http://www.netcracker.com/