A Lightweight Toolchain to Validate, Visualize, Analyze, and Deploy ETSI NFV Topologies

Philippe Merle, Adja Ndeye Sylla, Meryem Ouzzif, Frédéric Klamm, Karine Guillouard

A Lightweight Toolchain to Validate, Visualize, Analyze, and Deploy ETSI NFV Topologies*

Philippe Merle*, Adja Ndeye Sylla†, Meryem Ouzzif†, Frédéric Klamm†, Karine Guillouard†
*Spirals project-team, Inria Lille - Nord Europe, Lille, France
Email: philippe.merle@inria.fr
†TGI/OLN, Orange Labs, Rennes, France
Email: {adjandeye.sylla, meryem.ouzzif, frederic.klamm, karine.guillouard}@orange.com

***Abstract*—We demonstrate a lightweight toolchain for validating descriptors of network functions compliant with the latest ETSI NFV standards, visualizing these descriptors in the form of various diagrams (*i.e.* network-oriented, function-oriented, and UML2-based), analyzing these descriptors formally with the Alloy Analyzer, and deploying these virtualized network functions on OpenStack.**

***Index Terms*—ETSI NFV, Validation, Visualization, Analysis, Deployment, OpenStack.**

I. Overview: Motivation and Propositions

Network function virtualization (NFV) [1] is the current trend for virtualizing network functions, particularly in 5G and IoT contexts, with the IT virtualization technologies. The *European Telecommunications Standards Institute* (ETSI) defines standards, called ETSI NFV, to foster the portability of virtual network functions (VNFs) and the interoperability between VNF platforms. In particular, ETSI NFV defines a standard data format [2] based on the OASIS's *TOSCA Simple Profile in YAML* [5] to describe VNFs. Several open source and commercial NFV platforms (*e.g.* ONAP, OPNFV, Cloudify, to name a few) should support the latest ETSI NFV standards soon. However, DevOps teams also need standalone/offline tools to understand, validate, analyze, and deploy their VNFs easily, *i.e.* without running a heavy NFV platform. To address this, we demonstrate a lightweight toolchain to 1) validate VNFs descriptors compliant with the latest ETSI NFV/OASIS TOSCA standards, 2) visualize these VNFs descriptors in the form of various diagrams (*i.e.* network-oriented, function-oriented, and UML2-based), 3) verify and analyze VNFs descriptors formally with the Alloy Analyzer [13], and 4) deploy virtualized network functions and services on OpenStack [14], a world-wide used open source cloud management tool.

II. Use-Case Network Service

To illustrate our toolchain, we use an early release of the Wireless Edge Factory (WEF) [6] developed by the IRT b<>com [7]. This version, previously called Universal Gateway (UGW) [8] and shown in Fig. 1, is a 4G virtualized core network framework to support multiple access types.

* This work is funded by Orange Labs in the context of the <I/O Lab> joint research laboratory. The authors would like to express their gratitude to the members of the IRT b<>com who provided the necessary source code and information of the UGW use-case, particularly Emmanuel Gouleau and Philippe Bertin.

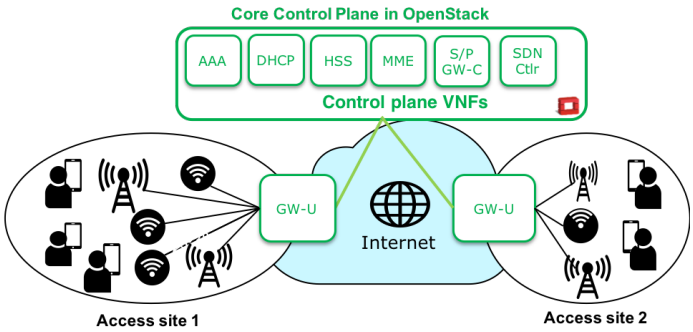

Fig. 1: UGW network service.

The core network control part is made of six network functions running in virtual machines:

- Authentication, Authorization & Accounting (AAA),
- Dynamic Host Control Protocol (DHCP),
- Home Subscriber Server (HSS),
- Mobility Management Entity (MME),
- Serving & PDN Gateway Control plane (S/PGW-C),
- SDN Controller (SDN-Ctlr) of S/PGW user plane (GW-U).

As input for our toolchain, we beforehand describe and package the UGW framework in compliance with the latest ETSI NFV standards [2]–[4]. Thus, the UGW is described as an ETSI compliant network service (NS) composed of the six VNFs aforementioned and one physical network function (*i.e.* PNF GW-U). The ultimate output of the toolchain is the UGW deployment using OpenStack.

III. Toolchain Architecture

Fig. 2 depicts the architecture of our ETSI NFV compliant toolchain providing four main features implemented by five components:

a) Validation: As input, the Parser component accepts *network service*, *virtual network function*, and *physical network function* descriptors (NSD, VNFD, and PNFD, respectively, as specified in [2]). The Parser component is set with ETSI NFV types (specified in [2]) and our own JSON Schema [9] of OASIS's *TOSCA Simple Profile in YAML* [5]. This schema allows us to validate NSD/VNFD/PNFD with respect to the syntax and typing of the TOSCA language. As

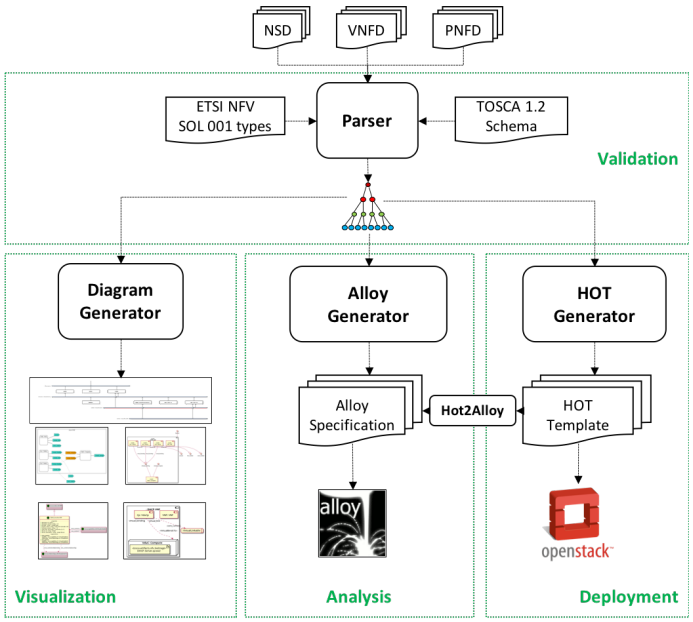

Fig. 2: Toolchain Architecture.

output, the PARSER component builds an in-memory hierarchical structure representing ETSI NFV descriptors, which is consumed by the generators described below.

b) Visualization: In order to better understand ETSI NFV descriptors, the DIAGRAM GENERATOR generates five kinds of visual diagrams: 1) *Network Diagram* represents the network interconnecting the different entities (see Fig. 3), 2) *TOSCA Diagram* represents how TOSCA nodes of network services/functions are interconnected throught their requirements and capabilities (see Fig. 4), 3) *UML2 Class Diagram* represents network service and function types (see Fig. 5), 4) *UML2 Component Diagram* gives a component-oriented view of network services/functions (see Fig. 6), and 5) *UML2 Deployment Diagram* gives a deployment-oriented view of network services/functions (see Fig. 7). We use the *nwdiag* tool [10] to generate network diagrams, the *dot* tool [11] to generate TOSCA diagrams, and the *PlantUML* tool [12] to generate UML2 diagrams.

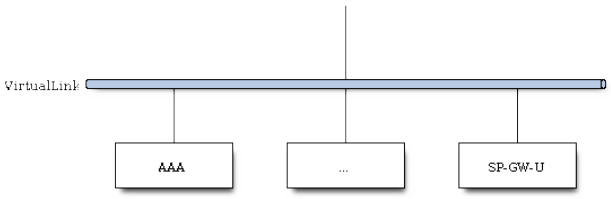

Fig. 3: Network Diagram for UGW NS.

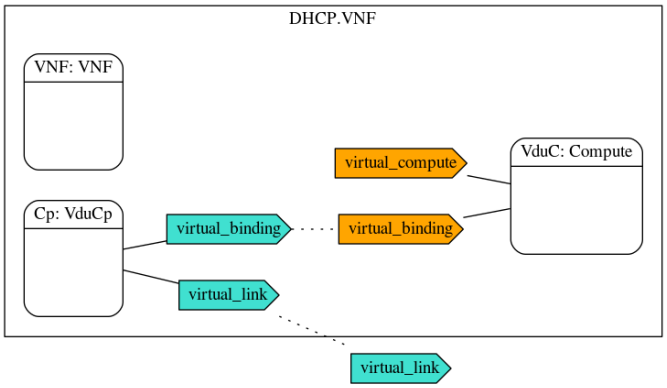

Fig. 4: TOSCA Diagram for DHCP VNF.

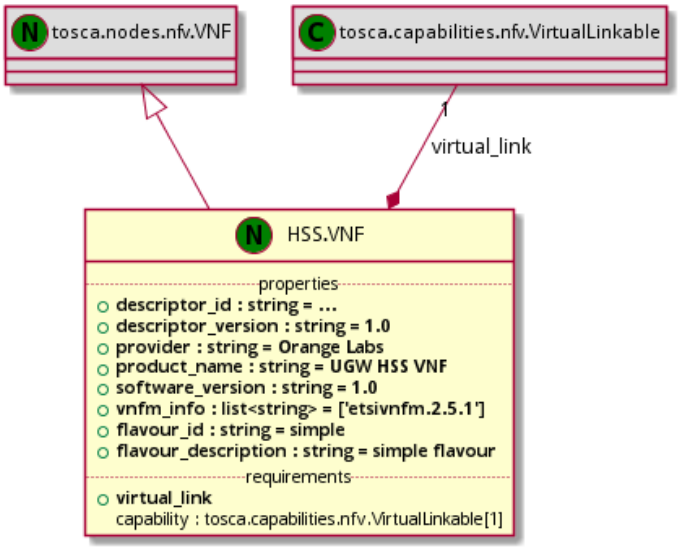

Fig. 5: UML2 Class Diagram for HSS VNF.

c) Analysis: In order to formally analyze network function/service descriptors, the ALLOY GENERATOR component generates Alloy specifications from ETSI NFV descriptors. Alloy is a lightweight formal specification language based on a first order relational logic [13]. Alloy specifications can be automatically analyzed with the Alloy Analyzer, a relational solver using SAT solvers. Then, DevOps teams can easily express and run various formal analysis on their ETSI NFV descriptors.

d) Deployment: The HOT GENERATOR component generates OpenStack Heat templates automatically. Then, the HOT2ALLOY component allows DevOps teams to translate these generated Heat templates into Alloy specifications in order to run Heat template specific analysis. Finally, DevOps teams can deploy the generated Heat templates on their OpenStack platform.

IV. PLANNED DEMONSTRATION

The demonstration consists in applying our ETSI NFV compliant toolchain on the UGW descriptors presented in

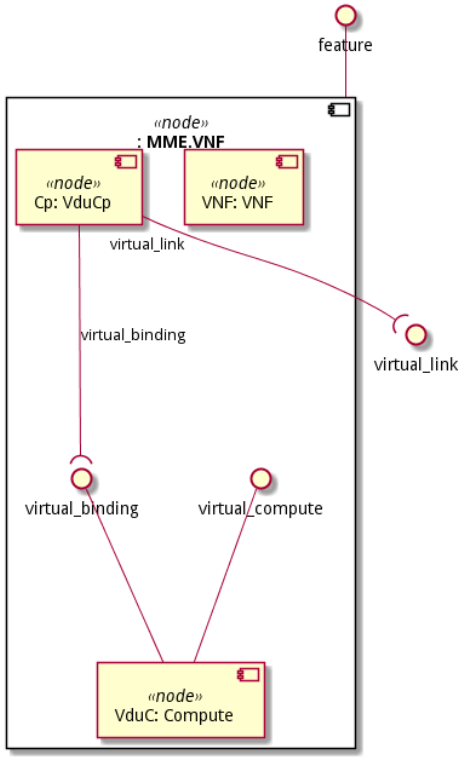

Fig. 6: UML2 Component Diagram for MME VNF.

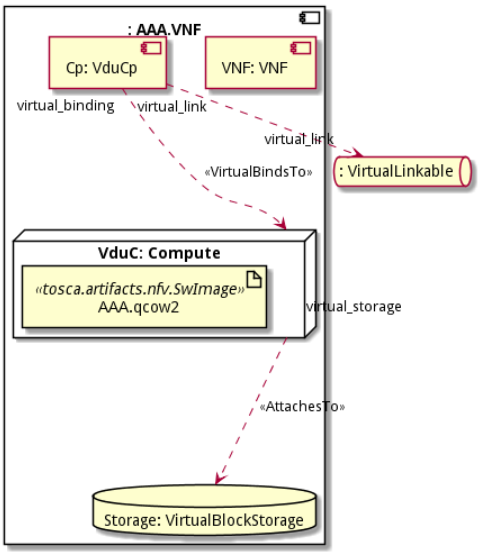

Fig. 7: UML2 Deployment Diagram for AAA VNF.

Section II:

1) We run validation of the UGW descriptors and show several examples of detected lexical/syntactic/typing errors.
2) We generate all the diagrams for the UGW use case and show the five kinds of supported diagrams, *i.e.* network diagrams (see Fig. 3), TOSCA diagrams (see Fig. 4), UML2 Class/Component/Deployment diagrams (see Fig. 5, 6, 7, respectively).
3) We generate Alloy specifications and run various analyses with the Alloy Analyzer.
4) We generate Heat templates from UGW descriptors.
5) We generate Alloy specifications from generated OpenStack Heat templates and run various Alloy-based analyses, for instance two virtual machines that are connected to the same port.
6) We deploy generated Heat templates on an OpenStack platform and show the deployed UGW service with the OpenStack dashboard.

V. Conclusion

We have demonstrated a lightweight toolchain compliant with the latest ETSI NFV standards [2]–[4]. This toolchain allows DevOps teams to validate ETSI NFV descriptors of their network services and functions, visualize their descriptors as network-oriented, TOSCA-oriented, and UML2-based diagrams, analyze their descriptors formally with the Alloy Analyzer [13], and deploy their network services and functions on OpenStack [14].

References

[1] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," IEEE Communications Surveys & Tutorials, 18(1):236-262, 2016.
[2] ETSI, "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification," ETSI GS NFV-SOL 001 v2.5.1, Dec. 2018.
[3] ETSI, "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification," ETSI GS NFV-SOL 004 v2.5.1, Sep. 2018.
[4] ETSI, "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; Network Service Descriptor File Structure Specification," ETSI GS NFV-SOL 007 v2.5.1, Dec. 2018.
[5] OASIS, "TOSCA Simple Profile in YAML Version 1.2," OASIS Standard, Jan. 2019.
[6] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, P. Bertin, A. Kerbellec, "On Evaluating Different Trends for Virtualized and SDN-ready Mobile Network," 6th IEEE International Conference on Cloud Networking (CloudNet 2017), Prague, Czech Republic, Sep. 2017.
[7] b<>com, "Wireless Edge Factory," 2019. [online] https://b-com.com/en/bcom-wireless-edge-factory.
[8] F. Guillemin, "Network function virtualization: some design and performance issues for network operators," Nov. 2017. [online] https://www.irt-systemx.fr/wp-content/uploads/2017/07/04-Fabrice-Guillemin.pdf.
[9] A. Wright, H. Andrews, "JSON Schema: A Media Type for Describing JSON Documents," IETF Internet Draft, Mar. 2018. [online] https://json-schema.org.
[10] T. Komiya, "nwdiag - simple network-diagram image generators," [online] http://blockdiag.com/en/nwdiag.
[11] E. R. Gansner, S.C. North, "An open graph visualization system and its applications to software engineering," Software - Practice and Experience, 30(11):1203-1233, 2000. Available at https://www.graphviz.org.
[12] "PlantUML," [online] http://plantuml.com.
[13] D. Jackson, "Software Abstractions: Logic, Language, and Analysis," MIT press, 2012.
[14] O. Sefraoui, M. Aissaoui, M. Eleuldj, "OpenStack: Toward an Open-Source Solution for Cloud Computing," International Journal of Computer Applications, 55(3):38-42, 2012. Available at https://www.openstack.org.