Modélisation des chaines de services hautement disponibles

par

Hind ERRAHMOUNI

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAITRISE
AVEC MÉMOIRE EN GÉNIE DES TECHNOLOGIES DE
L'INFORMATION
M. Sc. A.

MONTRÉAL, LE 17 DÉCEMBRE 2020

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Abdelouahed Gherbi, directeur du mémoire
Département de génie logiciel et TI à l'École de Technologie supérieure

Mme Nadjia Kara, codirectrice du mémoire
Département de génie logiciel et TI à l'École de Technologie supérieure

M. Alain Abran, président du jury
Département de génie logiciel et TI à l'École de Technologie supérieure

M. Chamseddine Talhi, membre évaluateur
Département de génie logiciel et TI à l'École de Technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 11 DÉCEMBRE 2020

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je tiens à remercier mon directeur de recherche, le professeur Abdelouahed Gherbi pour l'aide précieuse qu'il m'a apportée, pour son soutien continu et pour son œil critique qui m'a aidé à structurer mon travail.

Je tiens également à remercier ma codirectrice de recherche, la professeur Nadjia Kara pour sa disponibilité, l'appui scientifique qu'elle m'a fourni qui a permis l'amélioration de la qualité de mon travail.

Je tiens aussi à exprimer mes remerciements au président de jury, le professeur Alain Abran et au membre de jury évaluateur, le professeur Chamseddine Talhi.

Enfin, j'exprime mes remerciements les plus sincères à toute personne qui m'a aidé et qui a contribué à ce projet de recherche.

Modélisation des chaines de services hautement disponibles

Hind ERRAHMOUNI

RÉSUMÉ

La variété des plateformes de déploiement des services virtuels ainsi que la variété des modèles de description des services virtuels que ces plateformes utilisent peuvent créer une ambigüité pour les concepteurs de ces chaines de services. Ainsi le besoin d'un outil de description des services virtuels par les clients demandeurs et les orchestrateurs déclenche le besoin d'un modèle de description de service virtuels. Ce mémoire propose une solution venant du besoin d'avoir un modèle standard et générique regroupant les fonctionnalités de chaine de fonctions service et facilitant leur modélisation. Cette solution consiste à concevoir un modèle de description générique des SFCs permettant à la fois la conception graphique des chaines de fonctions service et leur adaptation aux plateformes de déploiements existantes. Le travail effectué dans ce mémoire est réalisé en suivant deux axes : le premier est la création du modèle de description générique des SFCs ainsi que l'éditeur permettant leur conception graphique. Le deuxième axe est l'implémentation d'outil de transformation permettant l'adaptation de ces SFCs vers les modèles de déploiements TOSCA et NFV. La validation de ce travail a été effectuée en se basant sur l'environnement d'implémentation et de test de l'éditeur et de l'outil de transformation.

Mots-clés : NFV, SFC, MDA, EMF, ATL

Highly available service chains modelling

Hind ERRAHMOUNI

ABSTRACT

The variety of virtual service deployment platforms as well as the variety of virtual service description models used by these platforms can create ambiguity for the service chains designers. Thus, the need for a virtual service description tool by service requesting clients and orchestrators triggers the need for a virtual service description model. This thesis proposes a solution coming from the need to have a standard and generic model grouping together the functionalities of the service function chain and facilitating their modelling. This solution consists of designing a generic description model of SFCs allowing both the graphic design of service functions chains and their adaptation to existing deployment platforms. The work done in this thesis is carried out along two axes: the first is the creation of the generic description model of SFCs as well as the editor allowing their graphic design. The second axis is the implementation of a transformation tool allowing the adaptation of these SFCs to the TOSCA and NFV deployment models. The validation of this work was carried out based on the implementation and testing environment of the editor and the transformation tool.

Keywords: NFV, SFC, MDA, EMF, ATL

TABLE DES MATIÈRES

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

NFV	Network Function Virtualization
SFC	Service Function Chain
VNF	Virtual Network Function
PNF	Physical Network Function
SF	Service Function
SFF	Service Function Forwarder
MDA	Model Driven Approach
PIM	Platform Independent Model
PSM	Platform Specific Model
CIM	Computational Independent Model
M2T	Model to Text
M2M	Model to Model
UML	Unified Modeling language
ATL	Atlas Transformation language
NAT	Network Address Translator
FW	Firewall
TM	Traffic Monitor
WOC	WAN Optimization Controller
IDPS	Intrusion Detection Prevention System
VOC	Video Optimization Controller

INTRODUCTION

0.1 Contexte

L'informatique des nuages ou le Cloud Computing gagne une attention signifiante dans le monde des ressources informatiques sous la forme de service et les centres de données virtuels deviennent de plus en plus populaires en tant qu'infrastructures rentables. Les services réseau évoluent d'un modèle centré sur les hôtes à un modèle centré sur les données tout en rapprochant les données et les ressources de calcul aux utilisateurs finaux. Pour répondre aux demandes dynamiques des utilisateurs, les opérateurs réseau ont choisi d'utiliser les ressources virtuelles qui peuvent s'adapter aux besoins applicatifs afin d'implémenter des services réseau sur un modèle physique statique et rigide. À l'avènement de la virtualisation des fonctions réseau (NFV : Network Function Virtualization), les instances des services réseau sont provisionnées à travers de multiples réseaux Clouds pour assurer la performance et l'équilibrage de charges 'load balancing'. L'interconnexion de ces instances pour former un service réseau complet de bout en bout est une tâche assez complexe, en raison du temps qu'elle nécessite et du cout élevé. (Deval B. et al., 2016)

Le travail de recherche effectué au cours de ce mémoire est élaboré dans le cadre de la modélisation des services virtuels. Ce mémoire de recherche se concentre sur l'identification des approches de modélisation des services virtuels qui peuvent être utiles dans la conception d'un modèle abstrait de chainage des fonctions service en étudiant sa capacité d'adaptation et de transformation en modèles spécifiques de plateformes de déploiements existantes.

0.2 Définition du problème

La problématique traitée par notre travail se manifeste dans la question suivante : comment arriver à modéliser un modèle abstrait des services virtuels et définir l'approche de transformation adéquate permettant l'adaptation de ce modèle aux plateformes de déploiement existantes ?

Afin de répondre à la problématique soulevée par cette recherche, trois étapes consécutives ont été identifiées :

- étape 1 : une revue de littérature visant en premier lieu la définition et l'identification des types et de techniques de la virtualisation. En deuxième lieu l'état de l'art des chaines de fonctions services et en troisième lieu la méthodologie dirigée par les modèles;
- étape 2 : le choix de la méthodologie la plus adéquate pour concevoir le modèle abstrait;
- étape 3 : la sélection de la meilleure approche de transformation à utiliser ainsi que l'approche de validation à suivre.

Le traitement des étapes précédentes permettra de soulever des problématiques secondaires. Ci-dessous une présentation détaillée de ces problématiques :

0.2.1 Quels sont les types et les techniques de virtualisation ?

Cette première étape correspond à l'identification et à la documentation des types et des techniques de virtualisation et en particulier la virtualisation des ressources et la virtualisation des fonctions réseau. Ce mémoire présente les aspects fondamentaux du domaine de la virtualisation, des modèles de spécification de services réseau virtuels existants et des méthodes de transformations modèle-au-modèle (M2M), et par la suite, le choix de la méthode de validation la plus adéquate sera effectué.

0.2.2 Quelle méthodologie est adéquate pour la conception du modèle abstrait ?

Au niveau de cette étape, il est primordial de déterminer les différents composants de notre modèle abstrait et de définir l'ensemble des données et champs relatifs à chaque élément du modèle. Des outils d'implémentation et de test du modèle devraient être choisis.

0.2.3 Quelle est l'approche de transformation à adopter ?

À la suite de la conception du modèle abstrait, le bon choix de l'approche de transformation permettra d'assurer une meilleure transformation vers les plateformes de déploiement cibles de notre étude. Il faut aussi choisir les outils d'implémentation et de validation adéquats.

0.3 Questions de recherche

Le présent mémoire offre une approche de modélisation abstraite de chaine des fonctions de services virtuels basée sur une étude des éléments qui le constituent. On traite également la possibilité de transformer ce modèle générique vers divers modèles de description de services utilisé par les plateformes de déploiements. Afin d'atteindre l'objectif de cette recherche, il faut répondre à ces questions :

1. Quels sont les composants d'une chaine de fonctions de services;
2. Quels sont les éléments et les données décrivant chacun de ces composants;
3. Quels sont les outils permettant l'implémentation et le test du modèle abstrait conçu à base de ces composants;
4. Quelle est l'approche de transformation la plus adéquate à appliquer à notre modèle;
5. Quels sont les outils permettant l'implémentation et quelles sont les métriques de validation définies de l'approche de transformation choisie.

La méthodologie suivie qui fournira les réponses à ces questions sera décrite à la section qui suit.

0.4 Méthodologie de recherche

La méthodologie suivie tout au long de ce travail est élaborée en se basant sur le cadre de Basili (Basili et Selby, 1991; Basili, Selby et Hutchens, 1986) adapté au domaine de génie logiciel. Ce projet de recherche a été établi en suivant les quatre phases consécutives présentées ci-dessous :

Phase de la définition : On commence par l'identification de la problématique puis par la documentation des questions de recherche. Par la suite, choisir la méthodologie de recherche en élaborant les étapes de conception de solutions, les scénarios de test ainsi que les méthodes de validation adéquates.

Phase de la planification : tout au long de cette phase, un ensemble de revues de littérature a été étudié portant sur les sujets suivants :

- les concepts de base de la virtualisation réseau;
- l'architecture de la chaine de fonction service et les étapes de chainage;
- l'infrastructure des fonctions de services NFVI;
- l'orchestrateur et gestionnaire des fonctions de services MANO;
- les langages et les modèles de description de chaine de fonctions service;
- les plateformes de déploiements des services virtuels;
- les différentes approches de transformations M2M.

Cette phase a pour but de déterminer les composants de la chaine de fonction service utiles pour la conception de notre modèle abstrait et le choix de l'approche de transformation appropriée.

Phase de développement et d'opération : cette phase tient en compte la configuration des outils utilisés pour la conception et l'implémentation du modèle abstrait de chaine de fonction de service ainsi que l'approche de transformation et la conception des scénarios de tests pour valider le modèle abstrait conçu et l'approche de transformation. Cet ensemble d'étapes mises en évidence est réfléchi à base de composants de la chaine de fonction service et l'approche de transformation choisie.
Phase d'interprétation : cette phase est dédiée à l'interprétation des résultats expérimentale et à un retour sur les questions de recherche initialement formulées.

L'interprétation se termine par l'identification des contributions apportées par cette recherche et des travaux futurs.

0.5 Organisation du rapport

Ce mémoire est organisé en quatre chapitres :

1. Dans le premier chapitre ayant comme titre « Les aspects fondamentaux liés à la virtualisation » nous introduisons les notions de base de la virtualisation;
2. Le deuxième chapitre intitulé « Revue de littérature sur l'architecture dirigée par les modèles et les approches de transformation » est dédié à la présentation de l'approche dirigée par les modèles et approches de transformations;
3. Le troisième chapitre ayant comme titre « Méthodologie de conception du modèle SFC » présente, comme son nom l'indique, la méthodologie de conception et d'implémentation du modèle abstrait de SFC et de l'approche de transformation;
4. Le dernier chapitre qui s'intitule « Test et validation du modèle SFC et des approches de transformations » se focalise sur la validation du modèle abstrait de SFC et à la validation de l'approche de transformation.

Vers la fin, nous présentons une conclusion mentionnant les contributions, les limites et les futures perspectives de travaux de recherche liés à notre travail.

CHAPITRE 1

LES ASPECTS FONDAMENTAUX LIÉS À LA VIRTUALISATION

1.1 Introduction

La virtualisation est un domaine de recherche qui a subi beaucoup d'évolution ces dernières années. Elle permet de fournir des services personnalisés aux utilisateurs de ressources sous forme de logiciel en découplant le traitement de réseau logiciel et les applications de leur matériel qui les supporte. (Yong Li, Min Chen, 2015)

Elle permet également l'amélioration les logiciels de gestion du réseau, la réduction des couts des équipements, et éviter les limites dues à la rigidité de la topologie physique. (Reuel Nathan Wandji Ngassam, 2018)

Ce premier chapitre effectue une synthèse de la présentation des aspects fondamentaux portant sur les types et techniques de la virtualisation en général, la virtualisation des fonctions réseau et le chainage de fonction service.

1.2 Les types de virtualisation

La virtualisation peut être appliquée sur plusieurs équipements, d'une simple mémoire allant jusqu'à une ressource de topologie réseau complexe. Il existe plusieurs types de virtualisation dépendamment du type d'hyperviseur. On note qu'un hyperviseur ou moniteur des machines virtuelles (VMM: Virtual Machine Manager) est une plateforme qui permet une machine physique de faire dérouler des machines virtuelles. Cette section présente une synthèse de la littérature ciblée sur les différents types de virtualisation. (Reuel Nathan Wandji Ngassam, 2018)

1.2.1 Hyperviseur type 1

Ce type d'hyperviseur (figure 1.1) est mis entre une couche logicielle directement au-dessus du matériel et elle le pilote directement. (Figure 1) Comme exemples de ce type d'hyperviseur, on cite Citrix XenServer et VMware ESX/ESXi. (Zhani Mohamed Faten, 2018)

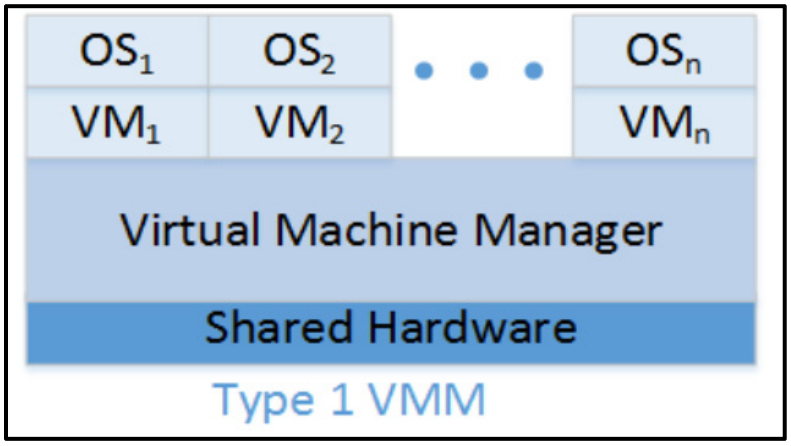


Figure 1.1 Architecture de l'hyperviseur type 1
Tirée de Zhani (2019)

1.2.2 Hyperviseur type 2

Ce type d'hyperviseur (figure 1.2) s'installe au-dessus du système d'exploitation. Comme exemples de ce type d'hyperviseur, on cite VMware Workstation, Oracle VM, Virtualbox et QEMU. L'hyperviseur type 1 offre une meilleure performance et sécurité que celui du type 2. (Zhani Mohamed Faten, 2018).

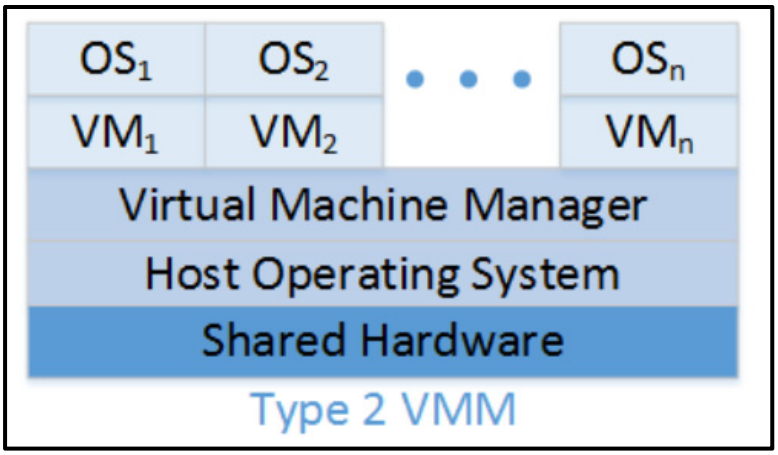


Figure 1.2 Architecture de l'hyperviseur type 2
Tirée de Zhani (2019)

1.3 Les techniques de virtualisation

1.3.1 Virtualisation complète

C'est une technique de 1re Génération où le système d'exploitation invité est non modifié et n'est pas conscient de la virtualisation. L'hyperviseur réécrit en temps réel toutes les instructions du système d'exploitation invité et simule le matériel sous-jacent. Cette technique permet une meilleure performance et isolation. On cite comme exemples de cette technique: VMware ESXi et Microsoft Hyper V Server. (Zhani Mohamed Faten, 2018)

1.3.2 Paravirtualisation

C'est une technique de 2e Génération qui permet une coopération entre l'hyperviseur et le système d'exploitation ou ce dernier est modifié et est conscient de la virtualisation. Cette technique ne supporte pas les systèmes d'exploitation non modifiés (par ex., Windows 2000/XP). Un exemple de cette technique, on cite: Xen. (Zhani Mohamed Faten, 2018)

1.3.3 Virtualisation assistée par matériel

C’est une technique de 3e Génération ou le système d’exploitation invité non modifié et n’est plus besoin de la réécriture binaire ou de la paravirtualisation. Cette technique permet au CPU d’offrir un support pour la virtualisation. Comme exemples de cette technique, on cite VMware ESXi et Xen. (Zhani Mohamed Faten, 2018)

1.4 Virtualisation des fonctions réseau

1.4.1 C'est quoi la virtualisation des fonctions réseau ?

Le terme NFV a initialement été proposé par plus qu’une vingtaine des plus larges opérateurs téléphoniques à savoir AT&T (American Telephone and Telegraph), BT (British Telecom) et DT (Deutsche Telekom). Selon l’Institut européen des normes de télécommunications (ETSI : European Telecommunications Standards Institut), l’idée de la virtualisation des fonctions réseau est reconnue comme une architecture réseau qui transforme la façon de construire et d’exploiter des réseaux en exploitant les technologies de virtualisation informatique standard et consolidant les fonctions réseau basées sur le matériel informatique propriétaire en des appareils standards (des machines d’architecture x86 par exemple). (Bo Yia et al., 2018)
La virtualisation des fonctions réseau (NFV) est une technologie réseau émergente. Au lieu de déployer des équipements matériels pour chaque fonction du réseau, les fonctions réseau virtualisées dans NFV sont réalisées via des machines virtuelles (VM) exécutant divers logiciels au-dessus des serveurs à haut volume ou de l’infrastructure cloud. La virtualisation des fonctions réseau permet de réduire les couts d’équipement matériel et la consommation d’énergie, améliorer l'efficacité des fonctions et optimise la configuration du réseau. (Wei Yang, Carol Fung, 2016).

1.4.2 Architecture de la NFV

L'architecture de la virtualisation des fonctions réseau (figure 1.3) se compose de l'ensemble des composants représentés dans la figure qui suit. En particulier, l'infrastructure de la NFV (NFVI) correspond sur le plan des données, qui transfert les données et fournit les ressources afin d'exécuter les services réseau. Management et orchestration réseau (MANO : Management and Network Orchestration) qui est un élément utilisé par la NFV pour gérer et accueillir des capacités que cette dernière a introduit pour la communication réseau, correspond au plan de contrôle. le plan de contrôle MANO est responsable de la construction des connexions entre les différentes VNFs (Virtual Network Functions : fonctions du réseau virtuelles) et l'orchestration des ressources dans l'infrastructure NFV. La couche VNF correspond au plan d'application, qui héberge des divers types des fonctions réseau virtuelles qui peuvent être considérées comme des applications. (Bo Yia et al., 2018)

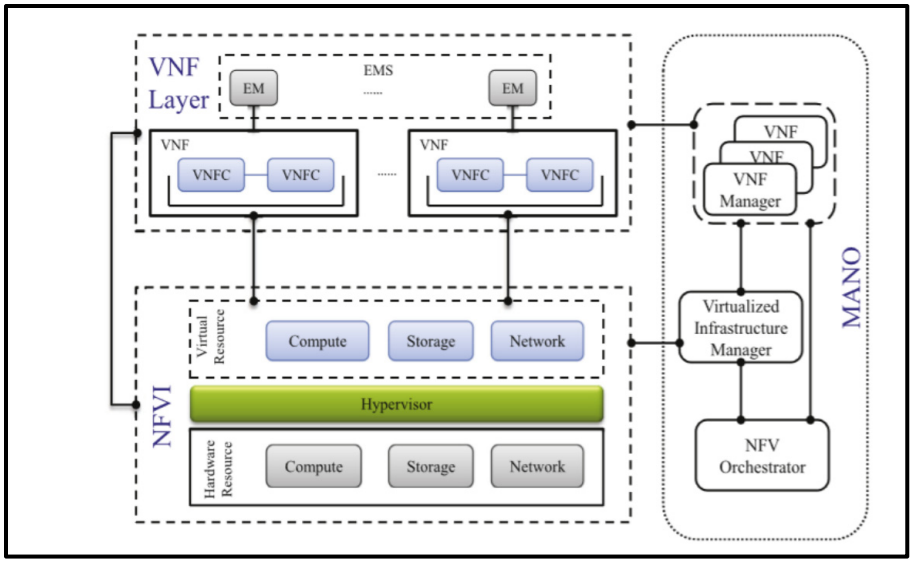


Figure 1.3 Architecture NFV de l'ETSI
Tirée de Bo Yia et al.(2018)

Couche NFVI (*Network Function Virtualization Infrastructure*):

NFVI fournit des services fondamentaux afin de remplir les objectifs de la NFV. En déployant un ensemble de périphériques réseau à usage général dans des emplacements distribués, l'infrastructure NFV peut satisfaire une variété des requis de service à savoir la latence et la localité, ainsi réduire le cout CAPEX et OPEX du réseau. Basée sur le matériel à usage général, NFVI fournit également un environnement de virtualisation pour le déploiement et l'exécution des VNF (fonctions virtuelles du réseau). Bien que les architectures des NFVIs actuelles sont généralement les mêmes, leurs implémentations peuvent beaucoup se différer. Selon la partie inférieure de la figure ci-dessus, l'architecture de référence de NFVI peut être divisée en trois couches distinctes : couche d'infrastructure physique, couche de virtualisation et couche d'infrastructure virtuelle. (Bo Yia, et al., 2018)

Couche d'infrastructure physique : l'infrastructure physique de la NFVI est composée de serveurs polyvalents qui fournissent les capacités de calcul et de stockage de base. En particulier, les serveurs qui fournissent la capacité de calcul sont nommés nœuds de calcul, tandis que ceux qui fournissent la capacité du stockage sont appelés nœuds de stockage. la communication entre les nœuds se fait à travers les interfaces internes. (Bo Yia, et al., 2018)

Couche de virtualisation : cette couche de NFVI se trouve entre l'infrastructure physique et l'infrastructure virtuelle. Ainsi, elle est considérée comme une plateforme logicielle qui exploite l'hyperviseur afin de séparer les ressources physiques et constituer un environnement isolé. Bien que tous ces environnements isolés partagent la même infrastructure, chacun d'eux est équipé avec tous les périphériques nécessaires d'une façon indépendante. En outre, beaucoup de comportements du réseau peuvent se produire dans l'environnement virtuel du réseau, comme l'instanciation, la suppression, migration en ligne et mise à l'échelle dynamique. Pour supporter ces comportements, l'hyperviseur peut ajuster d'une manière dynamique l'interopérabilité entre les ressources physiques et celles virtuelles allouées pour les machines virtuelles, de telle sorte à ce qu'une portabilité de haut niveau entre les machines

virtuelles puisse être atteinte. L'hyperviseur peut émuler presque chaque pièce de la plateforme matérielle. (Bo Yia, et al., 2018)

Couche d'infrastructure virtuelle : la couche d'infrastructure virtuelle se trouve au-dessus de la couche de virtualisation et contient trois types de ressources virtuelles : calcul, stockage et réseau. Ces ressources virtuelles jouent un rôle crucial pour fournir l'environnement de virtualisation dans la NFV. (Bo Yia, et al., 2018)

MANO : Management and Orchestration (Couche de management et d'orchestration de la NFV) :

Les responsabilités principales de MANO se reposent sur la gestion de l'ensemble du contexte virtualisé de l'architecture NFV. Le contexte comprend notamment un mécanisme de virtualisation, une orchestration des ressources matérielles, un management du cycle de vie des instances VNFs, etc. Selon l'organisation ETSI (l'Institut européen des normes de télécommunications), toutes ces responsabilités sont catégorisées en trois parties : VIM (Virtualized Infrastructure Manager), NFVO (NFV Orchestration) et VNFM (VNF Manager). (Bo Yia, et al., 2018)

Couche VNF (Virtual Network Function):
Cette couche joue un rôle important dans la structure de NFV tout entière. NFV est destinée d'abstraire les PNFs (les fonctions réseau physiques) sous-jacents et de les implémenter finalement dans la forme logicielle (VNF : fonctions réseau virtuelles).
Les VNFs peuvent fournir les fonctionnalités réseau qui sont originalement fournies par le propriétaire du matériel réseau, et devraient être exécutés sur le matériel informatique standard (COTS : commercial-on-the-shelf). La structure de la couche VNF peut contenir un grand nombre d'instances VNFs isolées. (Bo Yia et al., 2018)

1.4.3 Les requis techniques pour l’implémentation des VNFs

Dans cette section, nous résumons les exigences techniques lors de la mise en œuvre des VNF, y compris leurs performances réseau, leur facilité de gestion, leur fiabilité et leur sécurité.

La performance : lorsqu'on parle d'implémentation logicielle de fonctions réseau via des technologies de virtualisation sur des serveurs, la première question que l'on peut se poser est de savoir si les performances, telles que le débit et la latence, seront affectées.

La capacité par instance d'une VNF peut être inférieure à la version physique correspondante sur un matériel dédié.

Bien qu'il soit difficile d'éviter complètement la dégradation des performances, nous devons la garder aussi petite que possible, sans impacter la portabilité des VNFs sur les plates-formes matérielles hétérogènes.

Une solution possible consiste à exploiter les instances VNFs regroupées et les technologies logicielles modernes. Lors du déploiement des instances VNFs, il est nécessaire de concevoir des algorithmes efficients pour séparer la charge réseau sur un certain nombre de machines virtuelles distribuées et en grappes tout en gardant à l'esprit l’exigence de la latence. En outre, l'infrastructure NFV sous-jacente devrait être en mesure de collecter des informations sur les performances du réseau à différents niveaux (par exemple, hyperviseur, commutateur virtuel et adaptateur réseau). Lors de la conception des systèmes NFV, nous devons comprendre la performance maximale que peuvent atteindre les plates-formes matérielles programmables sous-jacentes. Sur la base de ces informations, nous pouvons prendre les décisions de conception appropriées. (Bo H. et Vijay G., 2015)

La maniabilité : l'infrastructure NFV L'infrastructure NFV doit être capable d'instancier les VNFs aux bons emplacements et au bon moment, allouer et échelonner dynamiquement les ressources matérielles et les interconnecter pour réaliser le chaînage des services. Cette flexibilité d’approvisionnement de service impose de nouvelles exigences en matière de

gestion des dispositifs existants et virtuels. La maniabilité de la NFV est assez différente de celle du réseau des centres de données, où les ressources matérielles sont presque équivalentes, ce qui facilite leur coordination. Cependant, le cout des ressources peut varier considérablement entre les points de présence du réseau et les locaux des clients. La fonctionnalité de gestion doit prendre en compte les variations et optimiser l'utilisation des ressources sur l'ensemble de la zone. Comme l'indisponibilité du service est généralement considérée comme inacceptable, les opérateurs réseau sur-provisionnent généralement leurs services ; ainsi l'utilisation des ressources allouées à ces services est normalement faible en raison de la redondance offerte pour une augmentation inattendue du trafic ou une défaillance de l'élément de service. Si nous partageons des ressources cloud via plusieurs services et que leurs modes de défaillance sont indépendants, nous pouvons tirer parti du des ressources disponibles pour assurer la redondance nécessaire entre eux et créer dynamiquement des VNFs pour gérer de manière appropriée l'augmentation ou la panne du trafic. NFV peut potentiellement améliorer l'utilisation des ressources grâce à la fonctionnalité de l'élasticité du Cloud, par exemple en consolidant la charge de travail sur un petit nombre de serveurs pendant les heures de nuit et en arrêtant le repos. La fonctionnalité de gestion devrait être en mesure de prendre en charge le partage des ressources de secours et l'approvisionnement des services réseau de manière efficace. Bien que NFV puisse rendre la maintenance planifiée relativement facile, elle présente de nouvelles exigences en matière de gestion de la qualité de service. Les opérateurs réseau doivent être capable d'obtenir et traiter des informations exploitables à partir de divers événements impactant les services, déterminer et corréler les défaillances et les récupérer en surveillant l'utilisation des ressources de calcul, de stockage et de réseau pendant le cycle de vie d'une VNF. Étant donné que les VNFs peuvent être créées / migrées d'une manière dynamique, ceci ajoute une dimension supplémentaire de complexité en ce qui concerne le suivi de l'exécution d'une VNF donnée. De plus, une VNF peut se comporter d'une manière irrégulière même si l'infrastructure sous-jacente fonctionne correctement, ce qui rend la détection des problèmes non négligeable. (Bo H. et Vijay G., 2015)

La fiabilité et la stabilité : la fiabilité est une exigence importante pour les opérateurs réseau lorsqu'ils proposent des services spécifiques (appels vocaux et vidéo à la demande, par

exemple), que ce soit par le biais d'appareils réseau physiques ou virtuels. Les opérateurs doivent garantir que la fiabilité du service et les accords de niveau de service (SLAs : Service Level Agreements) ne sont pas affectés lors de l'évolution vers la NFV. Pour répondre à la même exigence de fiabilité, NFV doit renforcer la résilience des logiciels lors du passage à des plates-formes matérielles. De plus, comme on en a déjà mentionné, l'élasticité de l'approvisionnement des services peut nécessiter la consolidation et la migration des VNFs en fonction de la charge de trafic et de la demande des utilisateurs. Toutes ces opérations créent de nouveaux points de défaillance qui doivent être gérés automatiquement. De plus, assurer la stabilité du service pose un autre défi à la NFV, en particulier lors de la reconfiguration ou du déplacement d'un grand nombre d'applications virtuelles logicielles provenant de différents fournisseurs et exécutées sur différents hyperviseurs. Les opérateurs réseau doivent être en mesure de déplacer les composants VNFs d'une plateforme matérielle vers une plate-forme différente, tout en respectant les exigences de continuité du service. Ils doivent également spécifier les valeurs de plusieurs indicateurs de performance clés pour assurer la stabilité et la continuité du service, notamment le taux de perte de paquets involontaire et le taux d'abandon des sessions / appels, le délai maximal par latence et la latence maximale. (Bo H. et Vijay G., 2015)

La sécurité : lors du déploiement de VNF, les opérateurs doivent s'assurer que les fonctionnalités de sécurité de leur réseau ne seront pas affectées. NFV peut apporter de nouveaux problèmes de sécurité avec ses avantages. Les machines virtuelles peuvent être exécutées dans des centres de données n'appartenant pas directement aux opérateurs réseau. Ces VNF peuvent même être sous-traités à des tiers. L'introduction de nouveaux éléments, tels que des orchestrateurs et des hyperviseurs, peut générer des vulnérabilités de sécurité supplémentaires qui augmentent la charge des systèmes de détection d'intrusion (IDS). Le réseau et le stockage partagés sous-jacents peuvent également introduire de nouvelles menaces de sécurité, par exemple lors de l'exécution d'un routeur logiciel sur une machine virtuelle qui partage les ressources physiques avec d'autres appareils réseau. De plus, ces composants logiciels peuvent être proposés par différents fournisseurs, créant potentiellement des failles de sécurité en raison de la complexité de l'intégration. Tous ces changements nous obligent à

repenser les problèmes de sécurité lors de la conception et de la construction des systèmes NFV. (Bo H. et Vijay G., 2015)

1.5 Chaînage de fonctions service

1.5.1 Définition d'une chaine de fonction services

Une chaine de fonction service définit un ensemble ordonné ou partiellement ordonné des fonctions services abstraits (SFs : Service Functions) et des contraintes organisées qui doivent être appliquées aux paquets, trames, et/ou les flux sélectionnés comme résultat de la classification. Un exemple de fonction service abstraite : un pare-feu. L'ordre implicite peut ne pas être une progression linéaire comme l'architecture le permet pour les SFC (Service Functions Chain) qui copie plus d'une seule branche, et qui permet aussi pour des cas où il y a une flexibilité dans lesquels les fonctions service ont besoin d'être appliquées. (Cisco Systems, 2015)

1.5.2 C'est quoi un chaînage de fonction services ?

Le chaînage des fonctions service est un mécanisme qui vise à fournir la capacité de définir une liste ordonnée des fonctions service (SFs : Service Functions) et diriger dynamiquement le trafic à travers des différents chemins des fonctions service. On a comme exemple de fonctions service : 'NAT: Network Address Translation', 'DPI : Deep Packet Inspection', 'FW : Firewall', traditionnellement, sont déployés sur des composants matériels. La SFC bénéficie pleinement de la combinaison de SDN/NFV en utilisant NFV (Network Function Virtualization) afin de virtualiser les fonctions service et de les connecter aux réseaux programmables dirigées par un contrôleur SDN (Software Defined Network) centralisé. Ceci fournit des déploiements automatisés et des opérations flexibles tout en permettant une implémentation forte et consistante de la sécurité ainsi d'autres politiques.(Van-Ca Nguyen, Anh-Vu Vu, 2017)

1.5.3 L'architecture du chaînage de fonction services

On décrit dans cette section les composants de l'architecture du chaînage des fonctions service. Dans la plupart des réseaux, les services sont construits en des séquences abstraites des fonctions service qui représentent les chaines de fonctions service (SFCs) (figure 1.4). Une SFC est une vue abstraite d'un service qui spécifie l'ensemble des SFs exigées dans le même ordre dans lesquelles doivent être exécutées. Les graphes, illustrés dans la figure ci-dessous, définissent une chaine de fonctions service, où chaque nœud du graphe représente au moins une fonction de service. Un nœud peut faire partie d'une ou plusieurs chaines de fonctions service ou pas, et un nœud donné peut apparaître une ou plusieurs fois dans une SFC donnée. Les SFCs commencent à partir du point d'origine ou bien de n'importe quel nœud suivant du graphe. Comme indiqué, les fonctions service peuvent donc devenir des nœuds de branchement dans le graphe, avec ces SFs on sélectionne les liens qui déplacent le trafic vers un ou plusieurs nœuds. Une chaine de fonctions service peut avoir plus qu'une terminaison. (Ericsson, 2015)

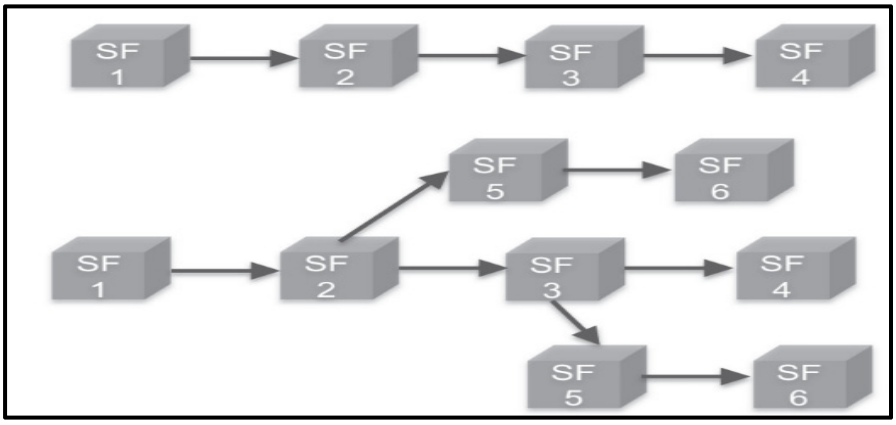


Figure 1.4 Service réseau, fonctions service et chaine de service
Tirée de Deval Bhamare et al. (2016)

Les chaines des fonctions service peuvent être unidirectionnelles ou bidirectionnelles. Une SFC unidirectionnelle exige que le trafic soit transmis à travers des fonctions service ordonnées dans une seule direction (exemple : SF1 >SF2 > SF3) alors qu'une bidirectionnelle nécessite un chemin symétrique (SF1 > SF2 > SF3 et SF3 > SF2 > SF1), et dans lesquelles les instances

SFs sont identiques dans le sens opposé. Dans une SFC hybride, on attribue les deux types de SFCs : unidirectionnelles et bidirectionnelles, c'est-à-dire quelques SFs demandent un trafic symétrique, tandis que d'autre ne traitent pas le trafic du sens inverse ou sont indépendantes du trafic correspondant transmis. SFCs peuvent contenir des cycles; dans ce cas, c'est possible que le trafic aura besoin de traverser une ou plusieurs fonctions de service dans plus qu'une seule chaine. (Ericsson, 2015)

Les chemins d'une fonction service : un chemin d'une fonction service (SFP : Service Function Path) s'agit d'un mécanisme utilisé par le chaînage de service afin d'exprimer le résultat du plus petit niveau de politique et des contraintes opérationnelles aux exigences abstraites d'une chaine de service. (Ericsson, 2015)

Les composants de base de l'architecture du SFC (figure 1.5) : une SFC est construite à partir d'un ensemble de composants logiques qui sont : les classificateurs, les transmetteurs des fonctions service (SFFs : Service Function Forwarders), les SFPs (Service Function Proxies) et bien évidemment les SFs. Tandis que l'architecture suivante représente ces composants logiques d'une manière fonctionnellement distincts, ils peuvent être réalisés en les combinant de diverses façons dans les produits déployés. (J Halpern Ed et C Pignataro Ed, 2015)

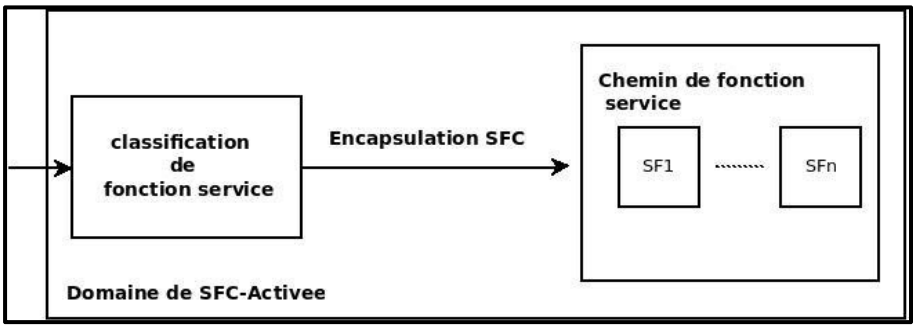


Figure 1.5 Architecture du chaînage de fonction service
Tirée de J Halpern Ed et C Pignataro Ed (2015)

Ce qui suit (figure 1.6), s'agit de l'architecture du SFC après la classification initiale (catégorisation des paquets en flux) et une explication des composants de l'architecture SFC ainsi qu'un aperçu détaillé de la façon dont certains de ces composants interagissent :

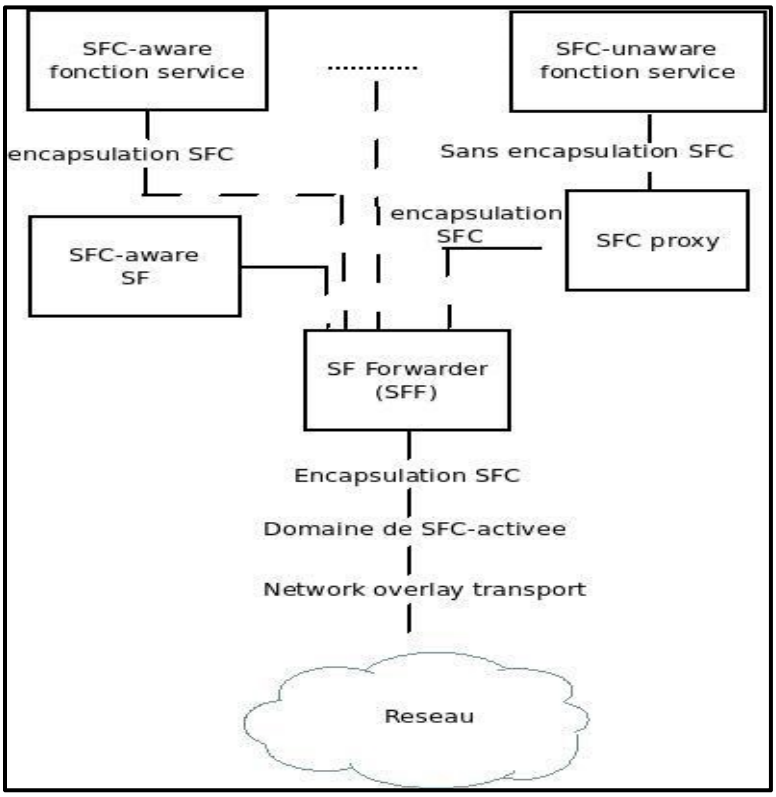


Figure 1.6 Architecture du chaînage de fonction service
Tirée de J Halpern Ed et C Pignataro Ed (2015)

Encapsulation du SFC : elle permet la sélection du chemin de la fonction de service ainsi le partage des métadonnées/informations liées au contexte lorsqu'un tel échange de métadonnées est requis. Elle contient des informations explicites utilisées afin d'identifier le SFP (Service Function Path). Pourtant, elle n'est pas une encapsulation de transport proprement dite : elle n'est pas utilisée pour transférer des paquets dans la structure réseau. Si les paquets doivent circuler entre des plateformes physiques distinctes, celle-ci repose sur un réseau de transport externe. Les re-dirigeurs de transit à savoir les commutateurs ou les routeurs, envoient les

paquets encapsulés avec l'encapsulation du SFC en fonction de l'encapsulation externe (non-SFC). L'un des principes clés de l'architecture de SFC est que l'encapsulation SFC reste indépendante du transport. À ce titre, tout protocole de transport réseau peut être utilisé pour acheminer le trafic encapsulé du SFC. (J Halpern Ed et C Pignataro Ed, 2015)

Fonction Service (SF) : le concept d'une SF évolue, Une fonction qui est responsable du traitement spécifique des paquets reçus. Une fonction de service peut agir sur différentes couches d'une pile de protocoles (par exemple, au niveau de la couche réseau ou d'autres couches OSI). En tant que composant logique, une fonction de service peut être réalisée comme un élément virtuel ou être intégrée dans un élément de réseau physique. Une ou plusieurs fonctions de service peuvent être intégrées dans le même élément de réseau. Plusieurs occurrences de la fonction de service peuvent exister dans le même domaine administratif. Les fonctions services envoient des données à/de un ou plusieurs SFFs. Les fonctions service de la 'SFC Aware' ou SFC consciente reçoivent ce trafic avec l'encapsulation du SFC. (Une fonction service peut être consciente de l'encapsulation SFC (c'est-à-dire qu'il reçoit et agit sur les informations de l'encapsulation SFC) ou non (auquel cas, les données transmises au SF ne contiennent pas l'encapsulation SFC)). Tant que l'architecture du SFC définit le concept et spécifie quelques caractéristiques de la nouvelle encapsulation qui est l'encapsulation du SFC ainsi que les différents composants logiques afin de construire les SFCs, Les implémentations SF existantes peuvent ne pas avoir la capacité d'agir ou de s'intégrer totalement à la nouvelle encapsulation SFC. Afin de fournir un mécanisme pour telles fonctions service pour participer dans cette architecture, on définit un proxy d'une fonction service. Le SFP (Service Function Proxy) joue le rôle de passerelle entre l'encapsulation SFC et les SFs de SFC-aware. (J Halpern Ed et C Pignataro Ed, 2015)

Transitaires des fonctions service (SFFs : Service Function Forwarders) : un SFF est responsable de la transmission des paquets ou/et trames reçus du réseau à une ou plusieurs SFs associées avec un SFF donné en utilisant les informations véhiculées dans l'encapsulation du SFC. Le trafic revient finalement au même SFF, qui est responsable d'injecter le trafic dans le

réseau. Le composant SFF a les responsabilités primaires suivantes : la transmission du SFP, terminaison des SFPs et le maintien de l'état du flux. (J Halpern Ed et C Pignataro Ed, 2015)

Domaine activé de SFC : un ensemble de fonctionnalités peuvent être nécessaires d'être appliquées dans les frontières d'un réseau activé de SFC, pour ne pas perdre des informations concernant le SFC à titre d'exemple. Ainsi, dans le contexte de l'utilisation du terme 'nœud' afin de référer généralement à une entité qui représente un ensemble de fonctions, un nœud de frontière SFC signifie un nœud qui connecte un seul domaine activé SFC à n'importe quel nœud se trouvant dans un autre domaine activé SFC ou bien un domaine qui est SFC non-consciente (SFC unaware). (J Halpern Ed et C Pignataro Ed, 2015)

Superposition réseau et les composants réseau : sous le transmetteur de fonctions service (SFF), on trouve des composants qui ont la responsabilité de réaliser la transmission du transport (superposition ou overlay). Ils ne consultent pas l'encapsulation SFC ou la charge utile interne pour effectuer cette transmission. Ils consultent uniquement l'encapsulation du transport externe pour la superposition. (J Halpern Ed et C Pignataro Ed, 2015)

Proxy SFC : pour que l'architecture SFC prenne en charge les SFs du SFC non-consciente, une fonction proxy peut être utilisée. Cette fonction se pose entre un SFF et un ou plusieurs SFs pour lesquelles le SFF redirige le trafic. Le proxy accepte tout d'abord les paquets du SFF de la part de la fonction service. Ensuite, il supprime l'encapsulation SFC, et utilise un circuit d'attachement local pour distribuer les paquets aux fonctions service de SFC non consciente. Il reçoit également les paquets en retour de la fonction service, réapplique l'encapsulation et les retourne au SFF afin de les traiter au long du chemin de la fonction service. En effet, le proxy semble faire partie d'une SF d'une SFC non-consciente. (J Halpern Ed et C Pignataro Ed, 2015)

1.6 Les étapes du chaînage des fonctions service

Afin de fournir Afin de fournir une chaine de services de manière optimale, il existe quatre étapes nécessaires à suivre qui sont conçues pour résoudre les problèmes qui se posent dans le déploiement automatique des chaines de fonctions de service dans le réseau: la description de chaîne de service (SC-D : Service Chain Description), la composition de chaîne de service (SC-C : Service Chain Composition), le placement de graphe de transfert des VNFs ; VNFs Forwarding graph (graphe de transfert des VNFs) (VNF-P : VNF Placement) et l'ordonnancement des VNFs (VNF-S : VNF Scheduling). (Cisco Systems, 2015)

1.6.1 Description de chaîne de service

La première étape est la description des services. Elle a pour but de définir ce qu'un service et comment va être utilisé. Elle inclut les interfaces du service ainsi que les contraintes techniques pour son utilisation. Ces informations sont nécessaires pour déployer les services et interagir entre eux. On décrit deux différents modèles de description des services virtuels utilisés par des plateformes de déploiement telles que OpenStack, ONAP (Open Network Automation Platform), Huawei SFC, etc, et des orchestrateurs des chaines de fonctions service comme OSM (Open Source MANO). (G. Mirjalily, Z. Luo, 2018)

1.6.1.1 Le modèle TOSCA

TOSCA (Topology and Orchestration Specification for Cloud Applications) est un modèle de données qui peut être utilisé pour créer des descriptions de données des applications et des infrastructures des services infonuagiques. Il peut être aussi utilisé pour définir les relations entre ces services ainsi que leur comportement opérationnel. Ceci peut s'effectuer indépendamment du fournisseur créant ce service ou l'infrastructure technologique utilisée pour le délivrer. TOSCA fait l'abstraction de configuration des données loin du matériel (hardware) et des services spécifiques pour rendre les services cloud plus interopérables et

portables et aussi pour permettre l'automatisation des réseaux SDN, en combinaison avec NFV et les environnements infonuagiques, pour simplifier l'orchestration des services de bout en bout. Ce modèle peut fournir une description déclarative de la topologie d'application pour un réseau ou un environnement infonuagique qui inclut tous ses composants, et qui peut inclure dans le besoin de l'équilibrage de charge (load balancing), réseautage et d'autres logiciels. Il propose une grammaire de description de modèles de service pour les applications cloud (figure 1.7) qui se composent de :

- gabarit de la topologie, et c'est la partie qui nous intéresse pour la description de chaîne de service, qui décrit les éléments nœud qui forment les nœuds de la chaîne, relation ou lien entre les nœuds et groupe des noeuds.

- plan qui décrit les actions et les séquences qui permettent le déploiement du Template de Topologie.

- types de nœuds, de relations et d'artefacts. (Marouen M. et al., 2016 ; Tobias B. et al., 2012)

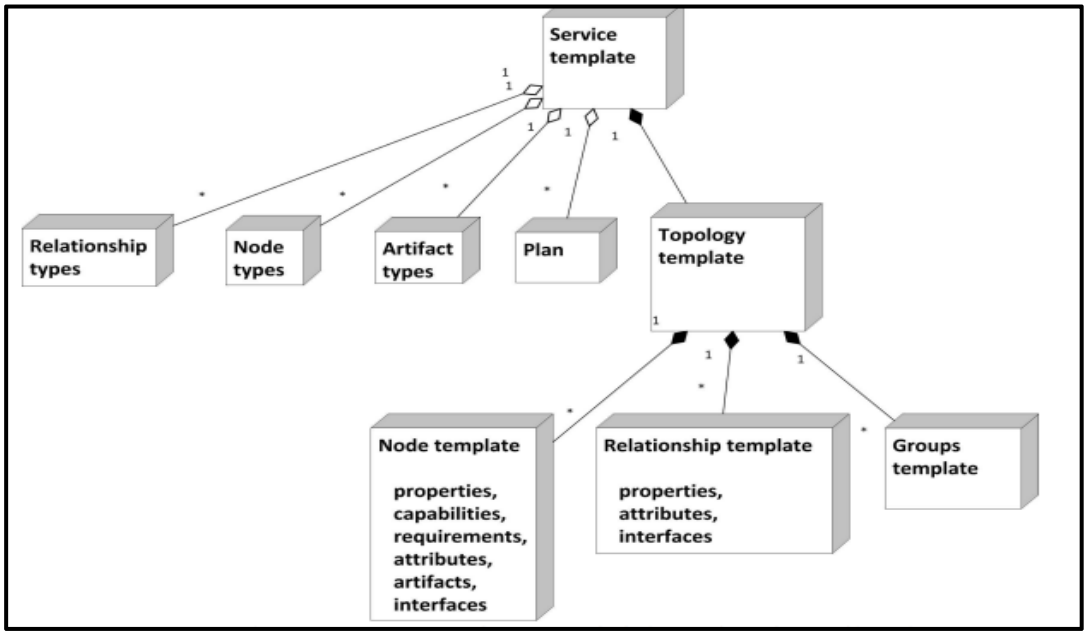


Figure 1.7 Le modèle de données TOSCA pour les applications cloud
Tirée de Marouen Mechtri et al. (2016)

1.6.1.2 Le modèle ETSI NFV

Le modèle de virtualisation des fonctions réseaux de l'ETSI (l'institut européen des normes de télécommunications) se compose de VNFD (Virtual Network Function Definition), PNFD (Physical Network Function Definition) et NSD (Network Service Description), entre autres, et comme ceci est illustré (figure 1.8) dans le diagramme d'arbre hiérarchique suivant:

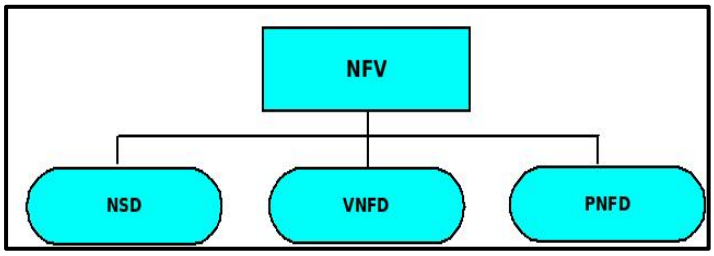


Figure 1.8 Le modèle Les descripteurs ETSI NFV
Tirée de ETSI-developers (2019)

Dans le travail de ce mémoire, on s'intéresse à cette première étape pour la description des chaines de fonctions service, plus précisément à la proposition d'un modèle de description de services virtuels.

1.6.2 Composition de chaîne de service

Dans le contexte de la NFV, un service réseau est défini comme une entité qui est composée d'une liste ordonnée des VNFs. La seconde étape dans le chaînage des fonctions service est comment concaténer les différents VNFs efficacement afin de composer un service réseau avec la méthode la plus appropriée et avec le respect des objectifs du demandeur. À titre d'exemple (figure 1.9), on considère une requête de chaîne de service avec un débit initial de données de 1Gbps et cinq VNFs : de VNF1 à VNF5. On suppose que VNF2 est un Load Balancer qui divise le flux de trafic entrant en deux flux égaux. Ainsi, on suppose que le débit du trafic sortant de VNF3 par rapport au trafic entrant est de 150% comme charge de trafic. L'ordre est lié aux dépendances entre les VNFs ; dans cet exemple, VNF2, VNF3 et VNF4

dépendent de VNF1 et donc elles doivent être exécutées après elle, et VNF3 doit être exécutée après VNF2. En se basant sur les dépendances, un nombre limité seulement des chaines de service sont possibles. La figure ci-dessous montre un exemple de deux compositions de chaîne de service possibles (deux graphes de transfert des VNFs). (G. Mirjalily, Z. Luo, 2018)

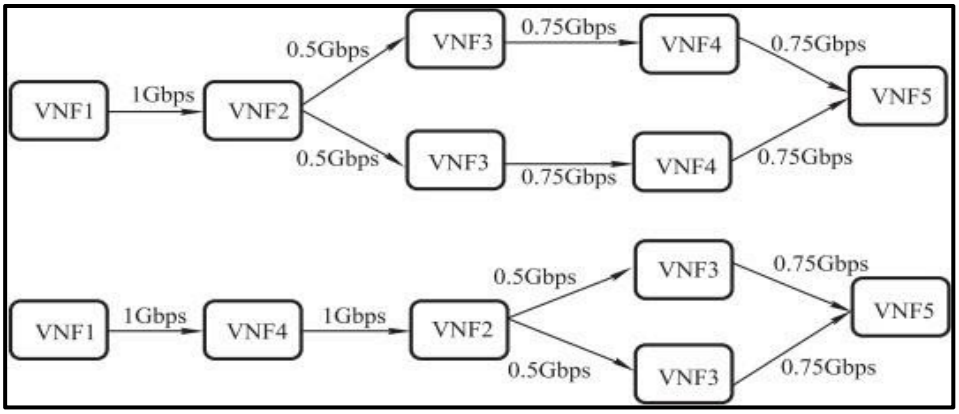


Figure 1.9 La composition d'une chaîne de service
Tirée de G. Mirjalily et Z. Luo (2018)

1.6.3 Placement du VNF-FG

Pour un ensemble des services réseaux demandés, le but est de placer les VNFs dans les serveurs physiques du réseau d'une manière optimale. Chaque VNF est définie par son type et ses ressources le calcul, le stockage et le réseau ; en effet, elle devrait être allouée dans le nœud physique qui a le même type. En pratique, l'orchestrateur s'appuyant sur les managers du VNF et le VIM (Virtualized infrastructure manager : gestionnaire de l'infrastructure virtualisé) évalue toutes les conditions pour assigner les VNFs aux ressources physiques. La plupart des travaux existants se focalisent sur l'étape de placement de la VNF(G. Mirjalily, Z. Luo, 2018). À titre d'exemple, Bari et al. (M. Bari et al., 2015) et Elias et al. (J. Elias et al., 2017)

1.6.4 Ordonnancement des VNFs

Les techniques d'ordonnancement permettent aux VNFs de partager les ressources afin de minimiser le temps d'exécution total des services réseau et bien d'autres. La figure suivante (figure 1.10) montre comment trois différentes chaines de service sont ordonnancées en cinq serveurs à haut volume HVS1-HVS5 pour minimiser le temps d'exécution total de l'ensemble du service. En tant qu'instance, le troisième service (S3) est composé de cinq fonctions réseau F1-F5, s'exécutant dans les serveurs HVS1, HVS4, HVS2, HVS5 et HVS1, respectivement. Si ces fonctions du réseau prennent 3, 1, 1, 3 et 2 unités de temps, respectivement ; ensuite le temps d'exécution total du service S3 est 10 unités de temps. (J.G. Herrera and J.F. Botero, 2016)

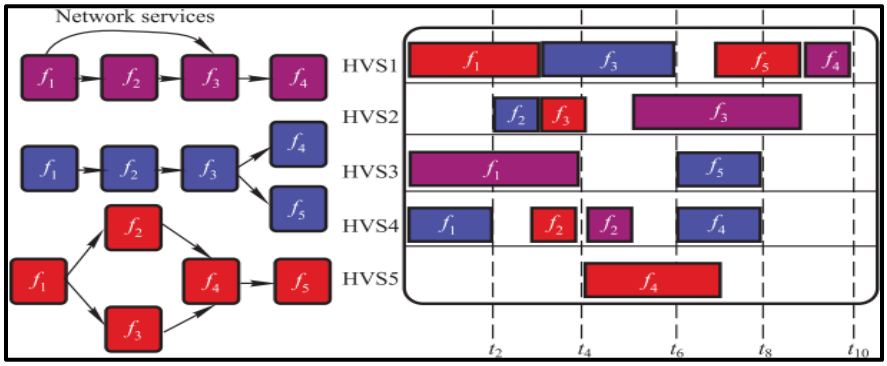


Figure 1.10 Ordonnancement des VNFs
Tirée de J.G. Herrera and J.F. Botero (2016)

1.7 Conclusion

Au cours de ce chapitre, on a parcouru les différents points de notre étude. Ce chapitre est débuté par un descriptif sur la virtualisation et en particulier la virtualisation des fonctions réseaux et le chainage de fonctions services. Notre étude se focalise sur la première étape de chainage des fonctions service afin d'élaborer le modèle de description des chaines de service en se basant sur l'approche dirigée par les modèles.

CHAPITRE 2

REVUE DE LITTÉRATURE SUR L'ARCHITECTURE DIRIGEE PAR LES MODELES ET LES APPROCHES DE TRANSFORMATIONS

2.1 Introduction

Dans ce chapitre, on présente les notions fondamentales de la modélisation dans le domaine de l'ingénierie logicielle qu'on s'est basé là-dessus pour effectuer notre modélisation. On considère donc les questions suivantes :

1. Qu'est-ce qu'une architecture dirigée par les modèles? ;
2. Qu'est-ce qu'une transformation de modèle?.

Dans cette étape de la recherche on a suivi deux axes dans notre revue de littéraires :

1. Une revue de littéraire sur les notions et les standards liés à l'architecture dirigée par les modèles;
2. Une revue de littéraire sur les approches et les outils de transformations de modèle.

2.2 L'approche dirigée par les modèles MDA

L'approche MDA a été présentée publiquement par l'OMG (Object Management Group, http://www.omg.org/) en Novembre 2000, elle constitue une déclinaison du MDE en réunissant l'ensemble des standards proposés par l'OMG. MDA est une approche qui se focalise sur l'utilisation des modèles pour la réalisation des systèmes informatiques, car elle part du principe que les modèles de conception sont pérennes dans le temps alors qu'une technologie de développement change constamment et peut rapidement devenir obsolète, les modèles deviennent alors des éléments productifs, contrairement à l'approche centrée sur le code, ou la conception du système ne représentait qu'une étape passagère pour arriver à la phase de l'implémentation, ce qui limitait le rôle des modèles à un rôle contemplatif et

purement descriptif été présentée publiquement par l'OMG (Object Management Group, http://www.omg.org/) en Novembre 2000, elle constitue une déclinaison du MDE en réunissant l'ensemble des standards proposés par l'OMG. MDA est une approche qui se focalise sur l'utilisation des modèles pour la réalisation des systèmes informatiques (figure 2.1), car elle part du principe que les modèles de conception sont pérennes dans le temps alors qu'une technologie de développement change constamment et peut rapidement devenir obsolète, les modèles deviennent alors des éléments productifs, contrairement à l'approche centrée sur le code, ou la conception du système ne représentait qu'une étape passagère pour arriver à la phase de l'implémentation, ce qui limitait le rôle des modèles à un rôle contemplatif et purement descriptif.(Abdelali Elmounadi, Naoual Berbiche, N Sefiani , 2013)

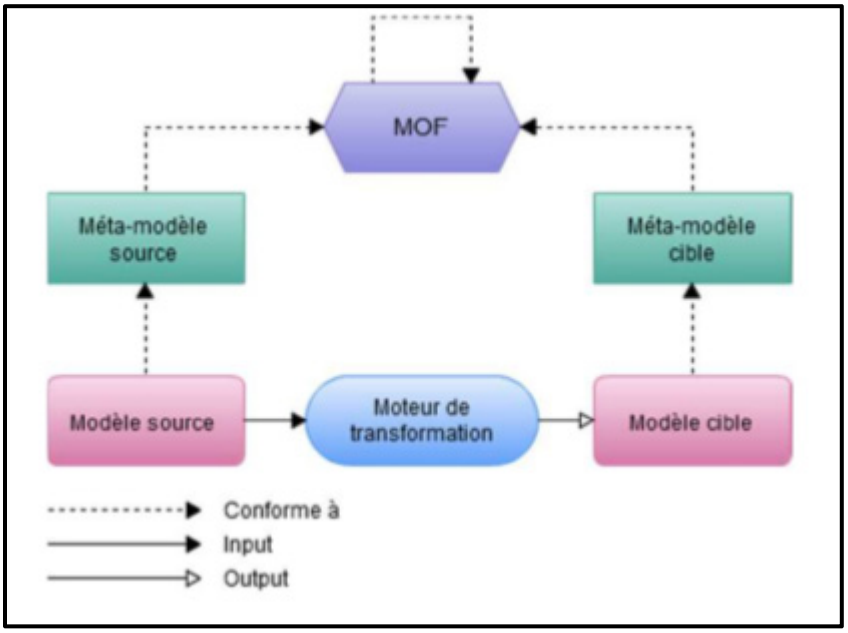


Figure 2.1 Aperçu global de l'approche MDA
Tirée de Abdelali Elmounadi et al. (2013)

L'approche MDA se base sur les trois notions suivantes (Ludovic MENET, 2010 ; Abdelali E. et al., 2013) :

- modèle,
- méta-modèle,

- transformation de modèle.

Un modèle est une abstraction de la réalité, il permet de représenter un système. Le modèle doit être conforme un autre modèle, d'où la notion de méta-modèle. Si l'on suit ce raisonnement, chaque modèle à un niveau d'abstraction donné doit être décrit par un modèle se trouvant au niveau d'abstraction supérieur, et on obtiendrait donc une infinité de niveaux hiérarchiques. Pour supporter cette approche, l'OMG propose le standard MOF [3] « Meta-Object Facility » qui permet de définir la représentation des méta-modèles et leur manipulation et qui a la particularité de se décrire réflexivement (figure 2.2). De cette façon, l'architecture dirigée par les modèles dispose de 4 couches de méta-modélisation :

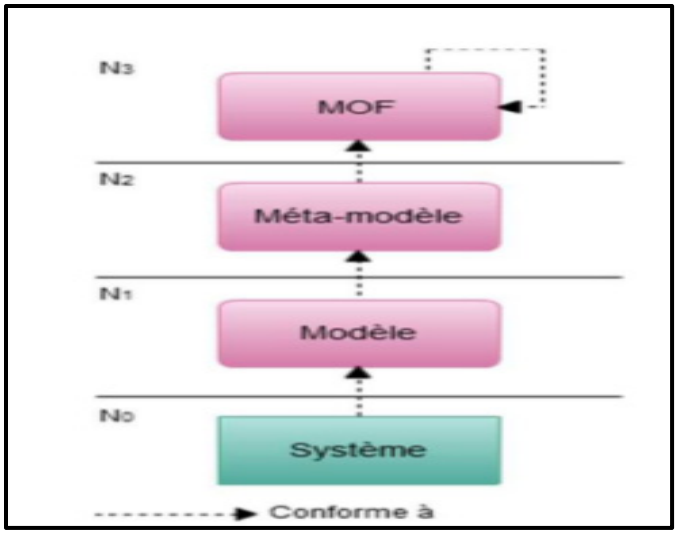


Figure 2.2 L'architecture à 4 niveaux de MDA
Tirée de Abdelali Elmounadi et al. (2013)

MOF est alors décrit comme un « méta-méta-modèle ». Étant donné que l'objectif principal de l'approche MDA est la séparation entre les préoccupations métiers et les contraintes techniques, celle-ci repose essentiellement sur les modèles suivants (figure 2.3) :

PIM (Platform Independent Model) : il s'agit des modèles d'analyse et de conception qui sont réalisés sans prendre en considération l'architecture technique de l'application.

PSM (Platform Specific Model) : il s'agit du modèle contenant les informations relatives à l'exploitation de la plate-forme d'exécution.

CIM (Computational Independent Model) : est un modèle qui définit la phase d'expression de besoin, qui décrit les exigences du système et qui est établi avant le passage au PIM.

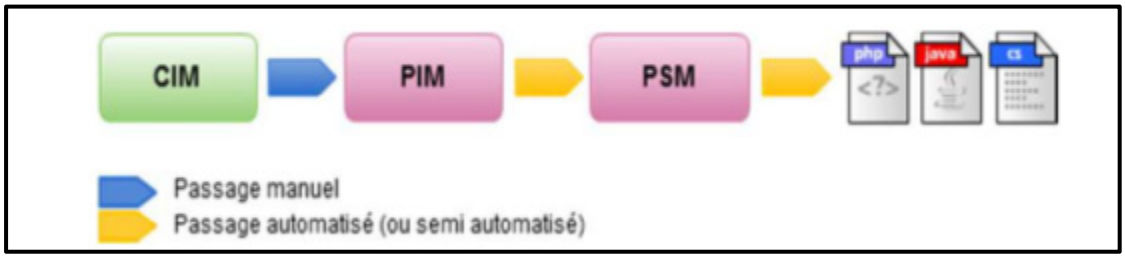


Figure 2.3 Les modèles intervenants dans l'approche MDA
Tirée de Abdelali Elmounadi et al. (2013)

La transformation de modèles (figure 2.4) dans un cadre MDA consiste alors à transformer les modèles PIM en des modèles PSM. Elle est réalisée grâce à un moteur de transformation qui applique un ensemble de règles au PIM fourni en entrée pour produire le PSM en sortie. La notion de méta-modèle est omniprésente dans ce cas, dans la mesure où chaque modèle (PIM ou PSM) s'appuie sur un méta-modèle qui sert à le décrire. Lorsque les deux modèles s'appuient sur le même méta-modèle, il s'agit alors d'une transformation « endogène », dans le cas contraire, nous parlons d'une transformation « exogène ». (Abdelali Elmounadi, Naoual Berbiche, N Sefiani, 2013)

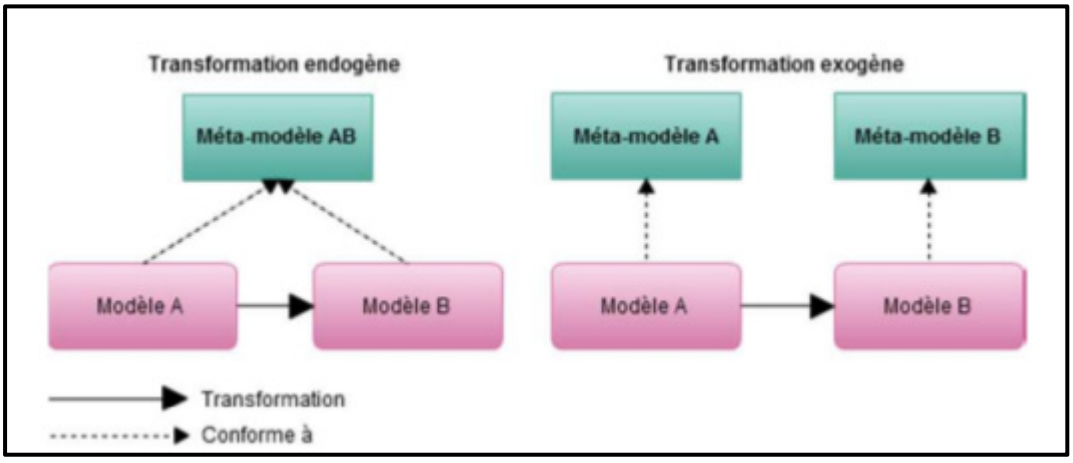


Figure 2.4 Transformation endogène et exogène
Tirée de Abdelali Elmounadi et al. (2013)

MDA s'appuie fondamentalement sur un ensemble de standards proposés par l'OMG à l'instar de MOF. Ces standards ont évolué pour pouvoir s'enrichir mutuellement :

UML (Unified Modeling Language) : le langage de modélisation unifié est un langage standard largement utilisé pour définir les exigences, analyser et concevoir et d'expliquer l'architecture de la programmation orientée objet dans le monde industriel. UML est un langage de modélisation informatique et communication visuelle qui utilise des diagrammes et un support texte. L'utilisation de UML n'est donc pas limitée à une seule méthode, mais UML est principalement utilisé en méthodologies orientées objet. (Bhakti Helvi Rambe et al. , 2020)

QVT (Query/View/Transformation) : le Query, View, Transformation (QVT) est une spécification faite par l'OMG. Le QVT standardise la transformation du modèle en divisant les transformations de modèle en deux niveaux d'architecture: métamodèle des relations et métamodèle de base. Dans le métamodèle des relations, la transformation du modèle entre les modèles candidats est spécifié comme un ensemble de relations. Ces relations doivent tenir pour une transformation réussie du modèle. Le métamodèle de base est défini en utilisant une extension minimale de l'essential MOF (EMOF) et OCL. Une transformation dans le métamodèle de base est défini comme un ensemble de mapping. Les relations du métamodèle peut être transformé au métamodèle de base par un ensemble de règles de mapping. (J Dong et S Yang, 2006)

XMI (XML Metadata Interchange) : le format d'échange de métadonnées XML (XMI) spécifie un modèle d'échange d'informations ouvert destiné à donner aux développeurs travaillant avec la technologie oriente objet la possibilité d'échanger des données de programmation sur Internet de manière standardisée, apportant ainsi la cohérence et la compatibilité aux applications créées dans des environnements collaboratifs. XMI permet l'échange d'objets à partir de la fonction d'analyse et de conception d'objets de l'OMG. Ces objets sont dans la plupart des cas décrits en UML (Unified Modeling Language) et MOF (Meta Objects Facility). (IBM Software, 1998)

2.3 Transformations de modèles

Le passage du modèle PIM vers le modèle PSM peut être effectue à travers une transformation automatisée (ou semi-automatisée). Le même type de passage est également distingue du modèle PSM vers le code. Ces deux passages peuvent êtres effectues par l'un des deux types de transformations de modèles suivantes:

Les transformations M2M (Model to Model) : il s'agit La transformation M2M fait référence au processus de modification d'un modèle pour créer un nouveau modèle ou effectuer sa mise à jour. (I. Porres, 2003)

Généralement, la transformation du modèle peut être vue comme une opération qui prend les éléments de modèle existants, puis applique certaines lignes directrices ou des règles sur ce qui devrait être modifié et comment il le sera, par la suite produit un modèle entièrement nouveau contenant les éléments appropriés ou la version modifiée du modèle contenant les éléments (nouveaux / modifiés). Les transformations M2M impliquent le processus de passage de (PIM) vers d'autres PIM ou depuis PIM aux modèles (PSM). (M Stephan, A Stevenson, 2009)

Les transformations M2T (Model to Text) : la transformation de modèle en texte (M2T) est une opération importante de gestion de modèle utilisée pour l'implémentation de la génération de code et de la documentation, sa visualisation et son exploration. Malgré la création du langage de transformation MOF Model-To-Text (MOFM2T) en 2008 qui constitue l'une des parties du projet MDA, beaucoup d'autres langages M2T existent aujourd'hui. (Louis M. Rose et al., 2012)

Ces transformations de modèles sont distinguées par certaines caractéristiques dont on cite:

La Réversibilité : une transformation est réversible s'il y a une autre transformation qui peut annuler l'effet de la première en allant dans le sens inverse.

La Réutilisabilité : une transformation est réutilisable si on peut l'étendre afin de créer de nouvelles transformations de modèles.

L'ordonnancement des règles : la représentation des niveaux d'imbrication de la transformation sont impliqués lors de la détermination de l'ordre d'exécution des règles de celle-ci.

La Traçabilité : une transformation de modèles est traçable si on peut garder toutes les informations pertinentes des étapes et liens de correspondance des éléments du modèle source et cible. (Abdelali Elmounadi, Naoual Berbiche, N Sefiani, 2013)

2.3.1 Les approches de transformations de modèles

En raison de mieux élaborer les règles de transformations, on définit une variété d'approches de transformation de modèles. L'arbre des approches existantes (figure 2.5) qui permettent d'établir des transformations de type 'modèle au modèle' M2M ainsi 'modèle au texte' M2T est là dans la figure qui suit:

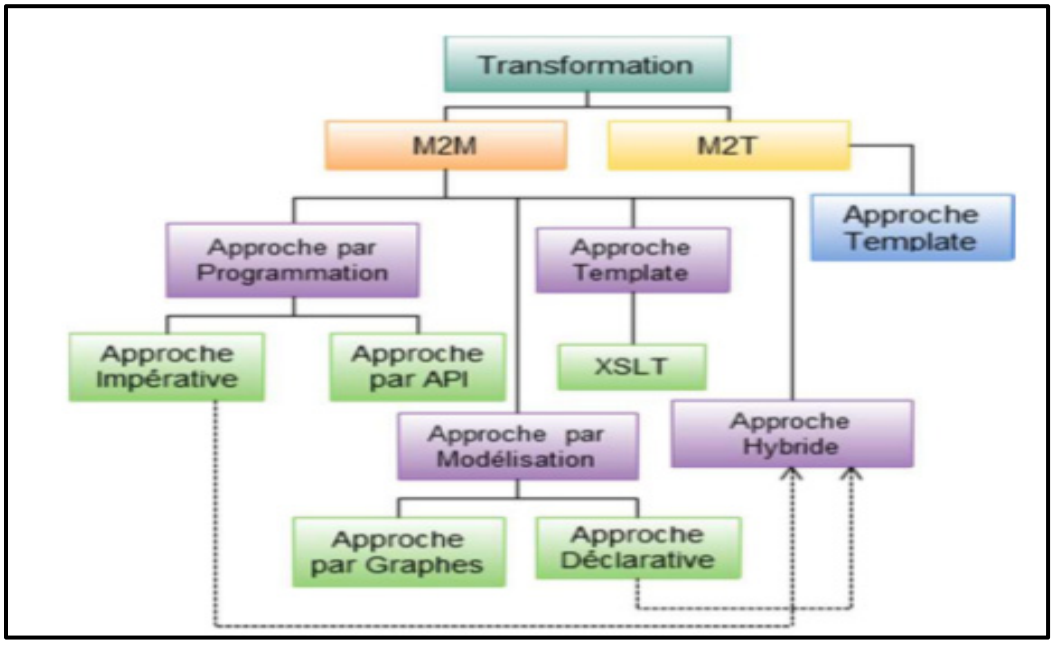


Figure 2.5 Relations des différentes approches de transformations de modèles
Tirée de Abdelali Elmounadi et al. (2013)

2.3.1.1 L’approche par programmation

Cette approche utilise les langages de programmation pour décrire la transformation, plus précisément les langages orientés objet. La représentation de cette approche est sous le format d'une application ou un programme informatique et reste la plus utilisée parmi les approches de transformations existantes. Il existe deux différents aspects a l'approche de transformation par programmation : impérative et par API. (Abdelali Elmounadi et al., 2013)

2.3.1.2 L’approche par template

Elle se base sur des patrons avec paramètres à valeurs dans les modèles sources afin de produire les modèle cible. L'implémentation de cette approche se référencie par l'approche XSLT avec le langage XLST (eXtensible Stylesheet Language Transformations). L'approche par Template s'adapte idéalement aux transformations de modèles car la conception du langage XSLT est basée sur la transformation des documents XML vers d'autres formats via XMI. Cependant, elle reste une approche moins performante et efficace par rapport d'autres dans le cas des modèles complexes. (Abdelali Elmounadi et al., 2013)

2.3.1.3 L’approche par modélisation

Elle consiste à modéliser les transformations par des règles suivant le concept mathématique 'théorie des graphes' en estimant celles-ci comme étant des graphes (approche par graphes). Par ailleurs, on peut modéliser les transformations par règles en faisant l'analogie entre les éléments du modèle source et ceux du modèle cible (figure 2.6) qui s'implémente par un moteur d'inférence (approche déclarative). L'approche de modélisation s'avère pénible en raison de la charge de travail que nécessite dans son développement. (Abdelali Elmounadi et al., 2013)

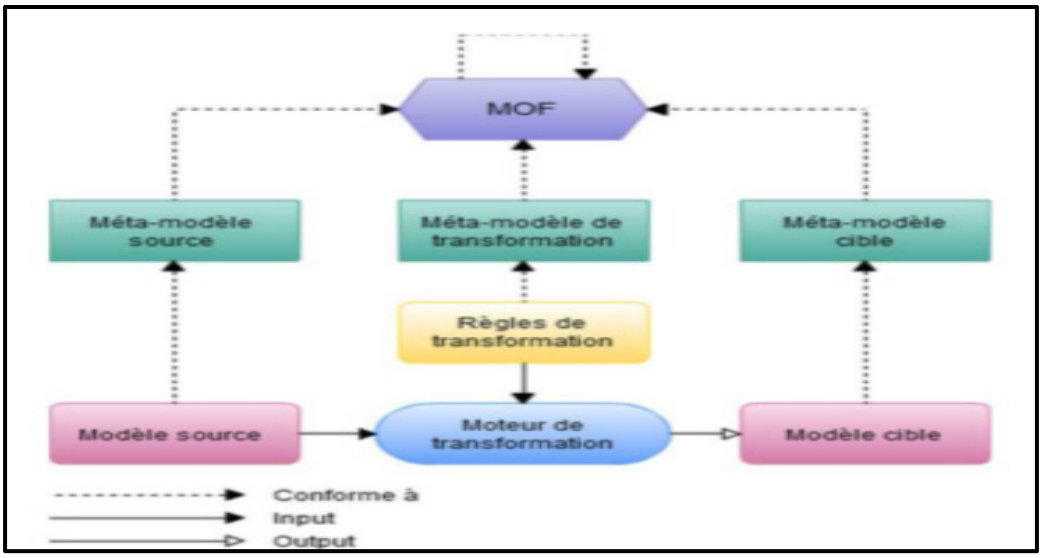


Figure 2.6 Cas de la transformation de modèle par approche de modélisation
Tirée de Abdelali Elmounadi et al. (2013)

2.3.1.4 L'approche hybride

Cette approche est la plus récente parmi les autres approches de transformation, elle combine l'approche déclarative afin de définir et sélectionner les transformations et impérative afin de décrire la stratégie de transformation. (Abdelali Elmounadi et al., 2013)

2.3.2 Outils de transformation de modèles

Après avoir parcouru les différentes approches de transformations, on définit les outils de transformations avec lesquels on pourra réaliser ces approches. Nous présentons uniquement les outils très connus open-source. (Abdelali Elmounadi et al., 2013)

1. AndroMDA : une plateforme qui s'adresse à générer le code des modèles UML. Cet outil utilise l'approche par Template pour la transformation de modèles. (Abdelali Elmounadi et al., 2013);

2. ATL : est un langage conçu dans le cadre du MDA pour coder les transformations de modèles utilisant l'approche hybride. Ce langage est déterminé comme un méta-modèle et ainsi comme une syntaxe textuelle concrète qui se repose sur l'OCL (Object Constraint Language) afin de fournir les règles de transformations. C'est un langage hybride adoptant l'approche déclarative le mieux adapté pour exprimer les transformations et impérative pour pouvoir exprimer les transformations très complexes. En effet, le code de transformation basé sur l'ATL s'agit d'un ensemble de règles les associations ou correspondances du modèle source avec le modèle cible/destination. (Abdelali Elmounadi et al., 2013);

3. Acceleo : est un outil générateur d'Eclipse. C'est un langage permettant l'implémentation des transformations de type modèle au texte [14] en adoptant l'approche par Template. Il exprime les éléments du modèle en s'inspirant de l'OCL aussi pour la navigation entre les composants du modèle ainsi extraire des données du modèle. (Abdelali Elmounadi et al., 2013);

4. Mola : 'MOdel transformation LAnguage' est un langage de transformation de modèles qui se base sur l'approche impérative afin de fournir les règles de transformation d'association des modèles. Il est également un langage de transformation graphique car il permet de concevoir un éditeur graphique. (Abdelali Elmounadi et al., 2013).

2.4 Conclusion

Dans ce chapitre nous avons présenté une description de l'approche dirigée par les modèles et des approches de transformations par modèles. Nous avons discuté aussi de différents outils de transformations utilisées. On a également répondu aux questions énoncées dans la section introduction de chapitre :

Qu’est-ce qu’une architecture dirigée par les modèles?

En répondant à cette question on a pu valider notre choix d’utiliser EMF comme outil de modélisation s’inspirant de MOF et supportant les concept clés de la MDA en utilisant les modèles comme entrée de développement et intégration des outils générant multiple langages de programmation (Java dans le cas de Eclipse EMF). Plus de détails sur l’utilisation de cet outil sont fourni dans le chapitre suivant.

Qu’est-ce qu’une transformation de modèle?

La réponse à cette question nous a permis de valider notre choix de type de transformation le plus adéquat, dans notre cas M2M, qui nous permettra de transformer notre modèle abstrait vers d’autres modèles existants ainsi que l’outil ATL utilisant l’approche hybride pour effectuer la transformation.

CHAPITRE 3

MÉTHODOLOGIE DE CONCEPTION DU MODÈLE SFC

3.1 Introduction

Notre recherche répond à la problématique suivante :

Comment concevoir un modèle SFC générique?
Comment concevoir la méthodologie de validation du modèle crée ?

Pour répondre à cette problématique nous avons analysé le chainage de fonction service afin de concevoir notre modèle et aussi on a mis en œuvre un environnement de validation et de test du modèle proposé. Ce dernier ainsi que l'environnement de test et de validation sont détaillées dans les sections qui suivent.

3.2 Modèle de description des chaines de fonctions service

Dans ce qui suit, on se focalise sur l'explication de modèle de données proposé qui a pour but de décrire d'une manière abstraite une chaîne de fonction service.

3.2.1 Diagramme de classe et choix et responsabilités des classes

Dans une section on présente le diagramme de classe (figure 3.1) du modèle ainsi, le choix et rôle des classes du diagramme pour justifier notre choix des données. Finalement, on décrit la première méthodologie suivie pour l'implémenter ainsi que les outils avec lesquels on réalise notre modèle de données.

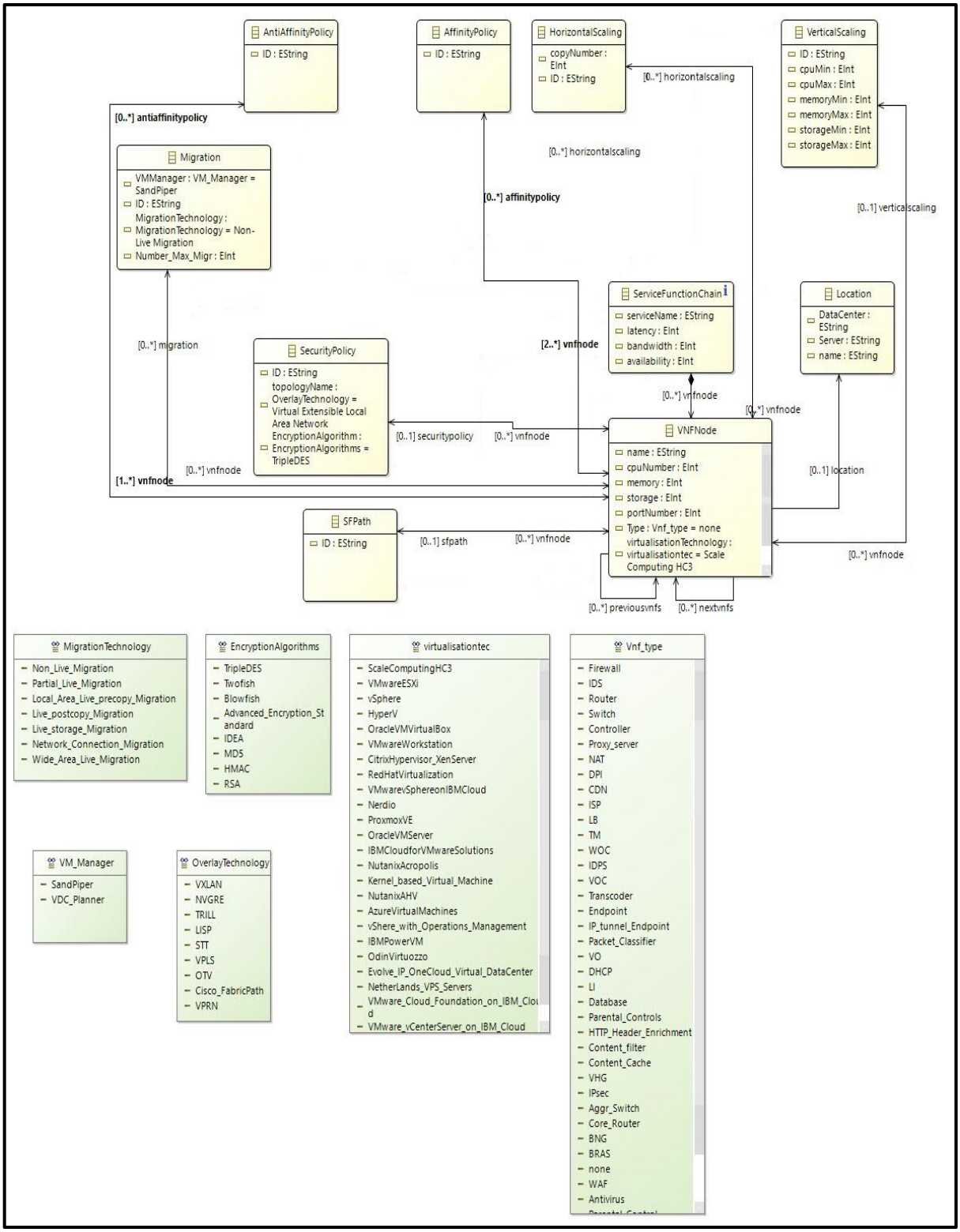


Figure 3.1 Diagramme de classes

Le diagramme de classe présente les informations de la chaîne de service (ServiceFunctionChain), les nœuds de la chaîne de service (VNFNode) ainsi que les politiques (Policies) liées aux nœuds de la chaîne de fonctions service. Location présente la localisation du nœud, AffinityPolicy est la règle d'affinité d'au moins deux nœuds qui ont la même localisation, AntiAffinityPolicy présente la règle d'anti-affinité qui exige que le nœud aura une localisation différente des autres nœuds, VerticalScaling décrit l'échelonnement vertical qui consiste à augmenter les ressources (cpu, mémoire, stockage) ou les diminuer selon le besoin, HorizontalScaling présente l'échelonnement horizontal qui permet d'avoir des copies du nœud afin de répartir les charges entre eux. Le diagramme présente aussi l'entité SecurityPolicy qui a les informations liées à la sécurité des nœuds/ de la chaîne de service, l'entité Migration qui permet de décrire comment une migration sera appliquée sur un nœud. Le tableau suivant (tableau 3.1) décrit les rôles des classes du modèle conçu.

Tableau 3.1 Choix et rôle des classes

Classes	Responsabilités	Dépendances
ServiceFunctionChain	représente les informations de la chaîne de fonction service	VNFNode
VNFNode	nœuds de la chaîne de fonction service	toutes les classes
Location	location du noeud	
AffinityPolicy	contrainte d'affinité des nœud; nœuds doivent être dans la même location	VNFNode
AntiAffinityPolicy	contrainte d'anti-affinité; nœud ou plusieurs doivent être dans une location différente	VNFNode
VerticalScaling	échelonnement vertical (Scale up/down)	VNFNode
HorizontalScaling	échelonnement horizontal (Scale out/in)	VNFNode
SecurityPolicy	représente les informations de la sécurité des nœuds/chaîne de fonctions service	VNFNode
Migration	représente les informations concernant la migration des nœuds	VNFNode
SFPath	représente les chemins de la chaîne de fonction service	VNFNode

3.3 Implémentation du modèle

Il s'agit de présenter la méthodologie choisie pour implémenter notre modèle de données décrivant les chaines de fonctions de service. Cette méthodologie consiste à réaliser un logiciel de création des chaines de fonctions à travers la description suggérée, qu'on appellera éditeur. L'idée d'implémentation du modèle s'opère par la mise en œuvre du diagramme de classe précédemment présenté et qui nécessite un ensemble d'outils qui ont pour but de déployer et donc valider le modèle des données proposé et précédemment décrit. Le diagramme de cas d'utilisation (figure 3.2), illustre les différentes fonctionnalités rendues par l'éditeur conçu:

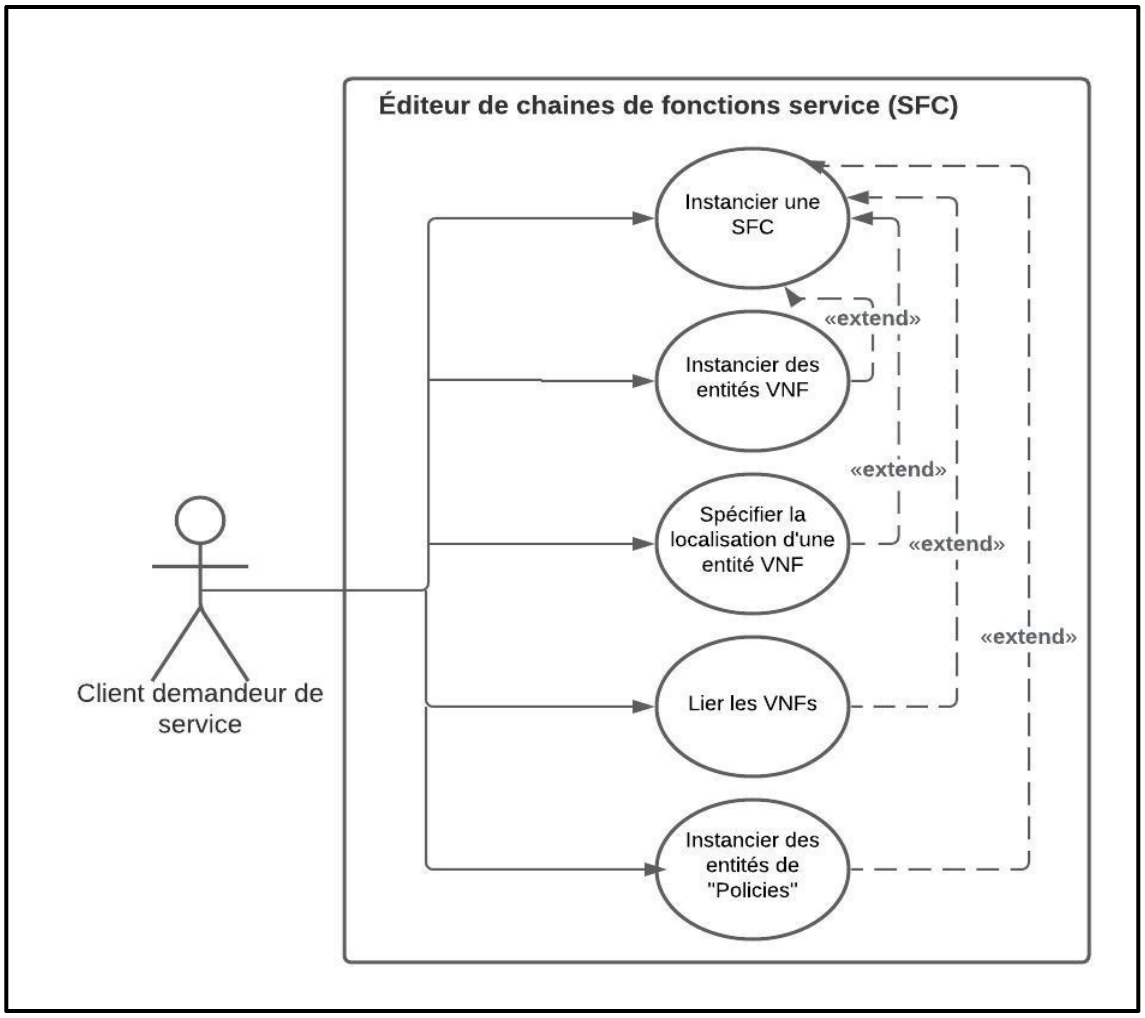


Figure 3.2 Diagramme de cas d'utilisation de l'éditeur SFC

3.3.1 Outils de modélisation

Les outils choisis pour l'implémentation du modèle et la réalisation de l'éditeur sont :

1. EMF (Eclipse Modeling Framework): l'EMF est un cadre de modélisation et un outil de génération de code pour créer des outils et des applications basées sur un modèle de données structurées, il nous permettra d'élaborer notre modèle de description des chaînes de service et de générer son code. Sur la base d'une spécification de modèle exprimée dans XMI, EMF fournit des outils et un support d'exécution pour produire un ensemble de classes Java pour le modèle, un ensemble de classes d'adaptateur qui permettent la visualisation et l'édition basée

sur les commandes du modèle, et un éditeur de base. Une caractéristique primordiale de l'EMF est qu'il fournit la base de l'interopérabilité avec d'autres outils et applications basés sur EMF. EMF se compose de trois parties fondamentales:

- EMF : le cadre EMF de base comprend un métamodèle (Ecore) pour décrire les modèles et le support d'exécution pour les modèles, y compris la notification de changement, le support de la persistance avec la sérialisation XMI par défaut et une API réfléchissante très efficace pour manipuler les objets EMF de manière générique.
- EMF.Edit : le Framework EMF.Edit comprend des classes réutilisables génériques pour la création d'éditeurs pour les modèles EMF.
- EMF.Codegen : la facilité de générer du code EMF est capable de générer tout le nécessaire pour construire un éditeur complet d'un modèle EMF. Il comprend une interface graphique à partir de laquelle des options de génération peuvent être spécifiées et des générateurs peuvent être appelés.il exploite le composant JDT (Java Development Tooling) d'Eclipse. (Jean Bézivin et al., 2004)

2**.** Sirius : un Framework Sirius (figure 3.3) est utilisé pour créer, visualiser et éditer des modèles à l'aide d'éditeurs interactifs appelés « modeleurs ». il va nous permettre de créer l'éditeur graphique des chaînes de fonction service à partir du code généré par EMF du modèle de description.

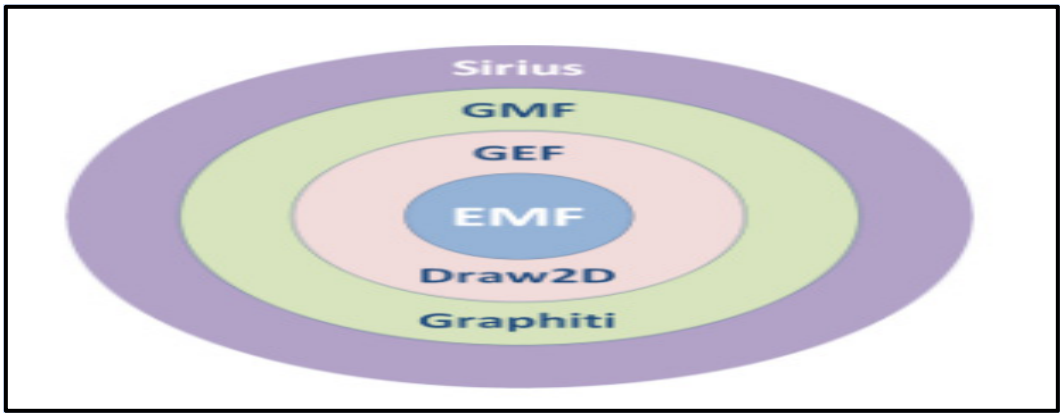


Figure 3.3 Hiérarchie des outils graphiques MDE (Model Driven Engineering)
Tirée de Vladimir Vujovic et al. (2014)

Sirius supporte trois types de représentations visuelles: les diagrammes (modélisateurs graphiques), les tableaux et les arbres (représentations hiérarchiques), mais de nouveaux dialectes peuvent être ajoutés par programmation.il offre la possibilité d'analyser, les rôles et les préoccupations des mêmes données en utilisant un point de vue différent sur le même modèle de domaine. Il fournit des outils pour spécifier les points de vue pertinents pour le domaine d'activité de l'utilisateur, quel qu'il soit. Parce qu'un Sirius utilise la spécification de domaine, qui n'est pas strictement dans le cadre de Sirius, il fournit un modeleur graphique pour la création d'un DSM (Domain-Specific Model), qui définit les concepts et leurs relations dans l'abstrait. Après avoir défini un modèle DSM, Sirius permet de créer facilement des représentations concrètes spécifiques de ces modèles, et les représentations peuvent être présentées dans plusieurs diagrammes, tableaux, matrices ou arbres. Les représentations ne sont pas statiques et complètent les environnements de modélisation dans lesquels l'utilisateur peut créer, modifier et valider ses designs. Il peut être organisé de manière logique en ‘viewpoints’, qui peuvent être activés ou désactivés par l'utilisateur final, dans le but de fournir une vue différente et cohérente sur le même modèle. Brièvement, Sirius simplifie le produit, réduit le temps de conception et augmente rapidement la productivité globale de la création d'un éditeur graphique spécifique au domaine. Les points forts de base de Sirius consistent en: la fondation sur une norme industrielle ouverte et largement utilisée - EMF, adaptabilité à tout DSM compatible EMF, une forte séparation entre les modèles sémantiques et de représentation, la prise en charge de différentes représentations de modèles de domaines, utilisation simple et développement rapide, le haut niveau d'extensibilité. Fourni avec le bon fichier de configuration, appelé un modèle de spécification de point de vue (VSM : Viewpoint Specification Model), qui décrit la structure, l'apparence et le comportement, Sirius peut représenter n'importe quel modèle compatible avec EMF.

Les cinq concepts principaux sur lesquels Sirius est basé sont stockés sous le VSM:

- Viewpoint : est un élément central qui est un ensemble logique de spécifications de représentation et de spécifications d'extension de représentation;

Représentation - est un groupe de construction graphique qui représente des données de domaine. Il décrit également la structure, l'apparence et le comportement des modèles;

- Mapping : dépend fortement du dialecte et identifie un sous-ensemble de l'élément du modèle sémantique qui doit apparaître dans une représentation et indique comment ils doivent être représentés;
- Style : est utilisé pour configurer l'apparence visuelle des éléments;
- Tool : décrit le mapping des comportements.

Les éléments d'un 'viewpoint' sont essentiellement la partie principale du modèle VSM et contiennent toutes les données des éléments de construction généralement décrits par des propriétés et des attributs. Certaines des propriétés sont décrites avec des noms de types qui sont généralement des types du modèle de domaine sémantique représenté. Un 'viewpoint' définit également des extensions de fichier modèle, qui restreignent la capacité de 'viewpoint' de modéliser uniquement les projets contenant des modèles sémantiques des types spécifiés. À l'intérieur d'un élément Viewpoint, il peut être créé:

descriptions des représentations, pour les diagrammes, les diagrammes de séquence, les tableaux et tableaux croisés, et les arbres, Règles de validation (basées sur un méta-modèle de règles de validation du modèle créé, elles seront appliquées à toutes les représentations définies à l'intérieur du point de vue), Extensions Java (définissent une classe Java qui contient une définition des méthodes de service). (Vladimir Vujovic et al., 2014)

3**.** Obeo designer: qui sera notre environnement de développement de l'éditeur graphique des chaînes de fonctions service. C'est un environnement de développement, appartenant au projet Open Source de la fondation Eclipse créé par Ober et Thales et propulsé par Eclipse Sirius, qui permet de créer facilement les propres éditeurs graphiques pour des domaines spécifiques (systèmes industriels, applications logicielles ou organisation de grandes entreprises). Les éditeurs créés avec Obeo Designer, comme la figure 3.4 l'illustre, sont composés d'un ensemble de représentations (diagrammes, tables et arborescences). (Obeo designer, 2020)

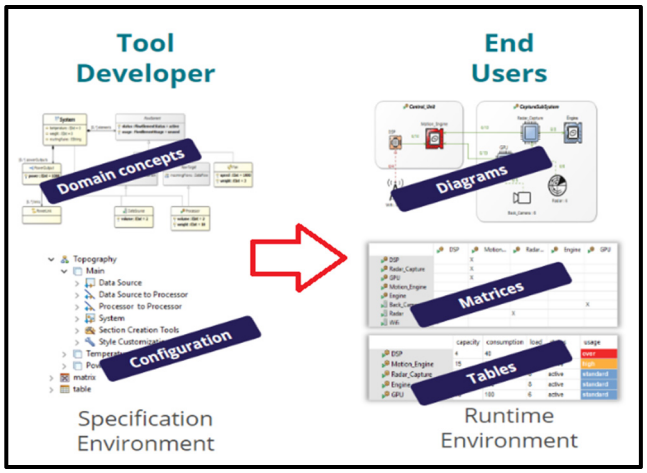


Figure 3.4 Les composants de Obeo Designer
Tirée de Sirius Obeo designer (2020)

3.3.2 Étapes d'implémentation

Le processus consiste en deux parties :

Création du méta-modèle de description des chaines de fonctions de service (figure 3.5) : le schéma suivant présente le processus d'implémentation de l'éditeur avec les outils précédemment définis en allant du développement du diagramme de classe de description du modèle jusqu'à la validation de ce dernier.

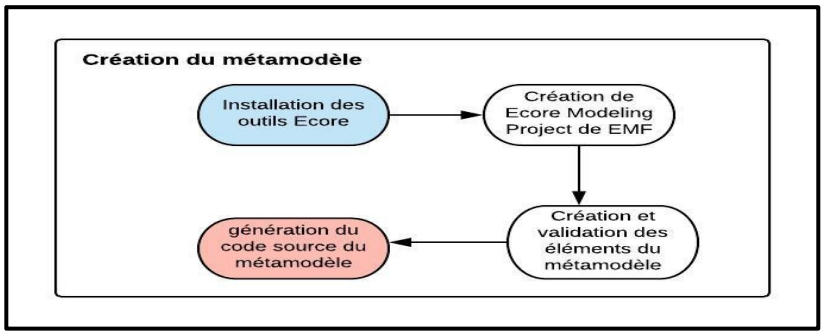


Figure 3.5 Les étapes d'implémentation du métamodèle SFC

En créant le projet, traçant le diagramme de classe (modèle de domaine), validant syntaxiquement et sémantiquement des éléments du méta-modèle et finalement la génération du code source Java du métamodèle qu'on exploite pour le reste du processus.

Test du méta-modèle (figure 3.6) :

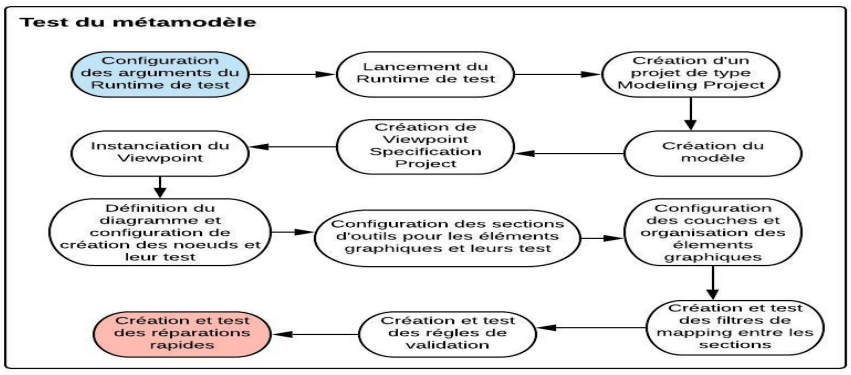


Figure 3.6 Les étapes de test du métamodèle SFC

En configurant les arguments d'une variable d'exécution 'runtime' de l'éditeur et lancement du 'runtime'. À cette étape on élabore le graphique de l'éditeur ainsi les contraintes améliorant la performance de l'éditeur indépendamment du méta-modèle en créant un projet de type 'modélisation'et en créant le modèle (qui se base sur le méta-modèle) ainsi qu'un autre projet de type spécification du point de vue 'viewpoint specification' qui contiendra le Workbench de modélisation. Ce qui suit consiste à l'instanciation du Viewpoint qui est l'ensemble des représentations (diagrammes, tables, arbres) que l'utilisateur final pourra instancier. On définit également le diagramme (compte tenu du fait qu'une chaîne de fonction service est un diagramme) et on configure la création des nœuds et leurs tests ainsi que des sections d'outils pour les éléments graphiques et leurs test et ceci en créant : les liens entres les nœuds (en codant des préconditions des liens) , la reconnexion des liens entre les nœuds , en configurant le suppression des éléments (nœuds et liens) et l'édition des labels des nœuds , en personnalisant des styles des éléments du diagramme (couleur, label, taille, etc). Par la suite, on configure des

couches des éléments graphiques et on organise ces éléments graphiques sur les couches (les couches représentent les différentes rubriques logiques dans la configuration des éléments graphiques, exemple : couche SFC contenant les éléments graphiques de la création d'une chaîne de fonctions service). On crée et on test des filtres de correspondance entre les sections graphiques de l'éditeur (exemple : filtre qui affiche uniquement les nœuds de la chaîne et cache tout autre entité du plan de travail). On termine la deuxième partie du test du méta modèle par la création et le test des règles de validation qui ont pour but de promouvoir la qualité de l'éditeur, et on finit par la création et test des réparations rapides pour corriger des problèmes dans la création de diagramme d'une manière automatique (exemple : créer un lien entre deux entités 'policies' alors que le lien consiste à lier les nœuds de la chaîne. On présente ci-dessous (figure 3.7) l'interface graphique de l'éditeur qu'on a nommé SFC. Elle contient le plan de travail, une palette des éléments de l'éditeur ainsi qu'un affichage des informations liées des entités au-dessous du plan de travail.

Figure 3.7 Composants de l'interface graphique de l'éditeur SFC

On a ajouté un ensemble de règles sémantiques dans notre implémentation, en utilisant le langage de requête Acceleo (AQL) qui sert à naviguer et interroger un modèle EMF (Acceleo Query Language, 2020) dont on cite:

- valeurs des attributs (figure 3.8): un intervalle de valeurs est spécifié pour chaque attribut entier permettant de borner les valeurs des propriétés;

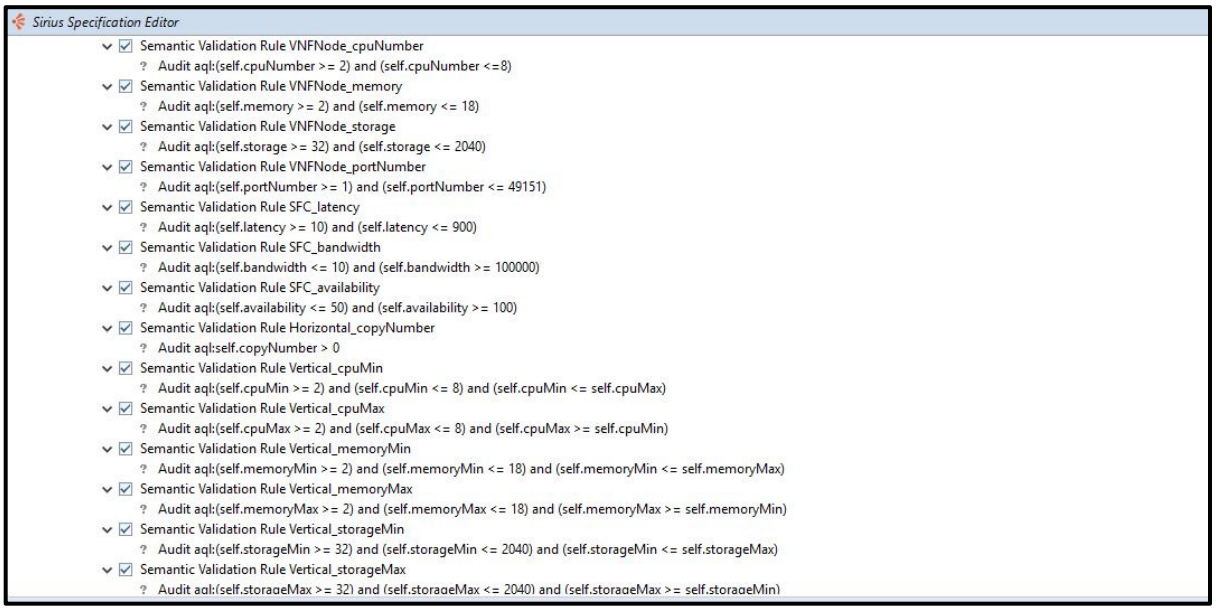


Figure 3.8 Règles appliquées sur les valeurs d'attributs entiers

- lien cyclique (figure 3.9): règle qui empêche les liens cycliques dans un nœud VNF;

- équilibrage de charges (Load Balancing) (figure 3.9) : règle qui vérifie le besoin d'un équilibreur de charge dans une chaîne de fonctions service non linéaire qui sert à répartir les charges du trafic dans le graphe tracé par l'utilisateur.

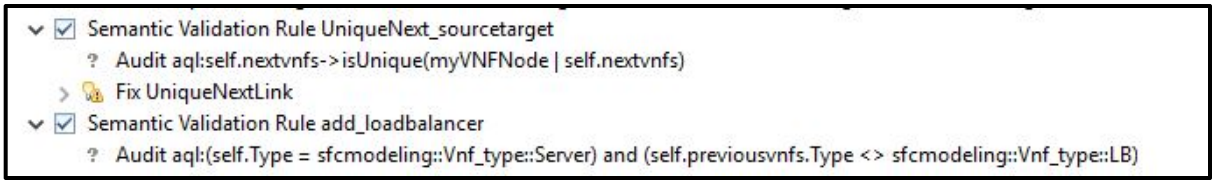


Figure 3.9 Règle du lien cyclique et de l'équilibrage de charge

3.4 Transformation du modèle de description de chaine de fonction service

Comme son nom l'indique, cette deuxième partie consiste à interpréter la transformation du modèle de description des chaines de fonctions service dont son diagramme de classe est précédemment présenté. On a choisi d'utiliser la transformation modèle-au-modèle (M2M) étant donné qu'on veut transformer les données de notre modèle vers un autre modèle, ce que le deuxième type de transformation modèle-au-texte (M2T) n'assure pas. Parmi les options offertes par la transformation M2M, on opte pour le langage ATL (Atlas Transformation Langage) qui se base sur l'approche la plus récente par rapports aux autres approches nommée hybride. Cette approche est une combinaison entre deux approches (déclarative et impérative) ce qui rend son adaptation au modèle assez performante en terme de sa puissance d'expression. On définit les trois termes (M2M, ATL, approche hybride) en détail dans le chapitre 2. Dans une section suivante, on montre l'implémentation des modèles de description des services virtuels cibles (TOSCA et NFV) dans le projet de transformation, on montre également la correspondance entre le modèle SFC source et ces deux modèles cibles ainsi qu'un exemple des règles de transformation développée dans le projet : transformation du modèle SFC vers le modèle de données TOSCA et transformation du modèle SFC vers le modèle de données NFV.

3.4.1 Correspondance des modèles

3.4.1.1 TOSCA

On présente le diagramme de classe (figure 3.10) du modèle TOSCA implémenté puis un schéma qui indique la correspondance des classes de notre diagramme vers celui de TOSCA (figure 3.11).

Le diagramme contient l'entité ServiceTemplate qui présente les informations liées au service virtuel, l'entité TopologyTemplate qui rassemble les éléments du service virtuel et contenant les informations liées à la topologie de la chaîne de service. Elle contient NodeTemplate qui représente les informations liées aux nœuds de la chaîne ainsi que Policy qui représente la politique de la chaîne.

On fait le mapping entre les classes de notre modèle SFC (les composants de la chaîne et les politiques) avec les classes du modèle Tosca comme suit : tout entité de notre modèle est un NodeTemplate et son type est spécifié par NodeTypes. Enfin, la classe ServiceFunctionChain correspond à ServiceTemplate.

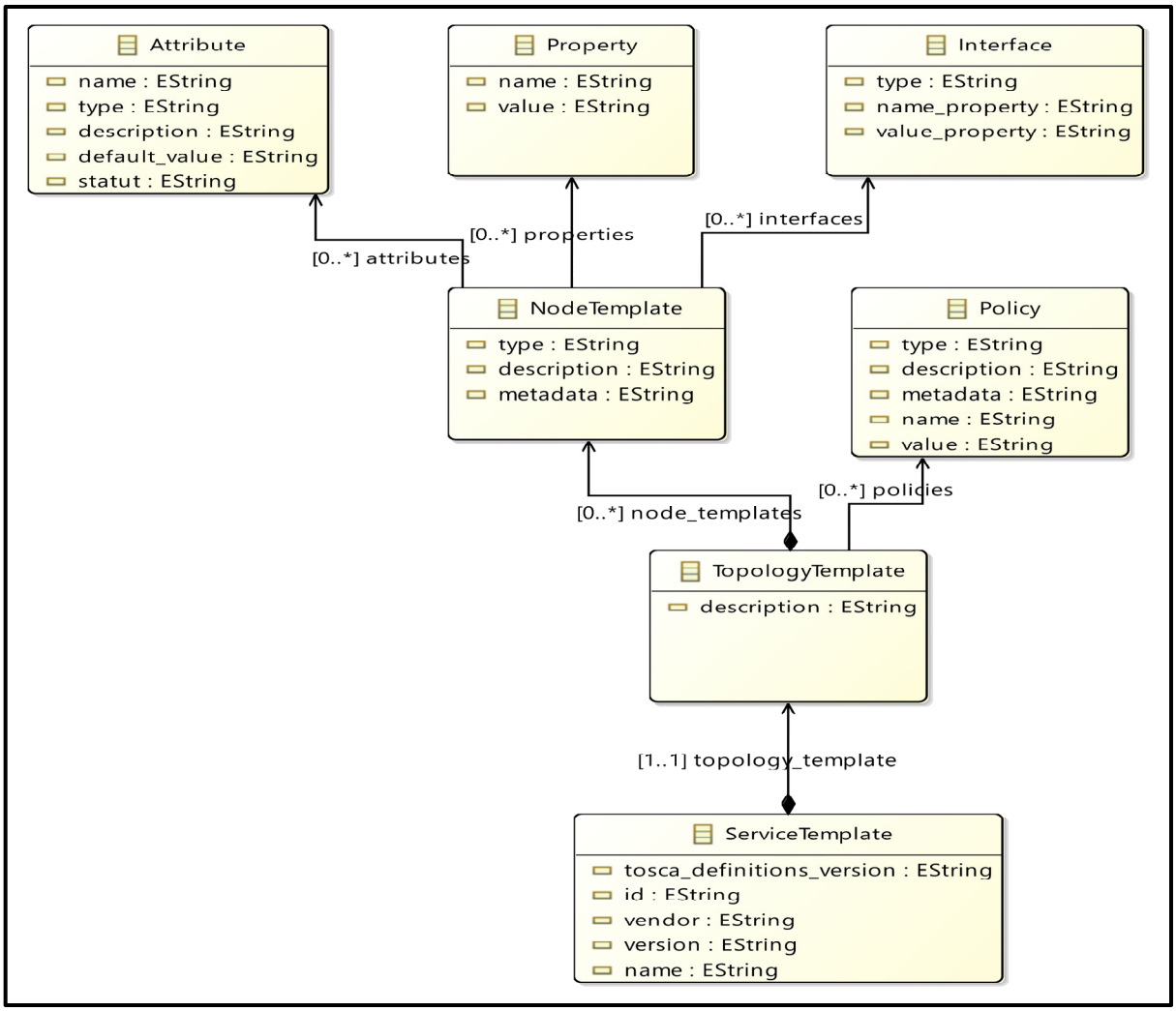


Figure 3.10 Diagramme de classe TOSCA

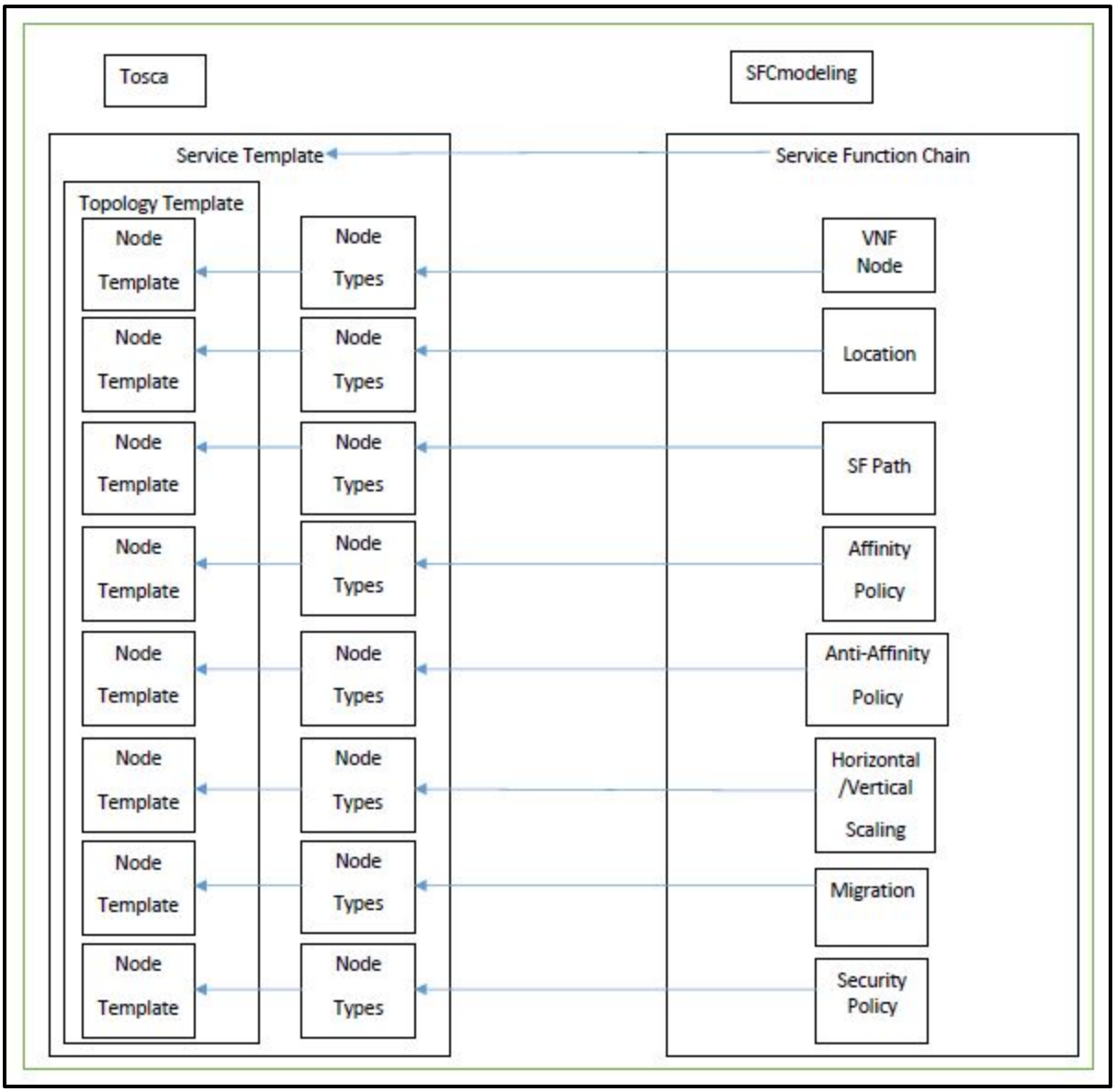


Figure 3.11 Schéma d'échange de données entre le modèle SFC et TOSCA

3.4.1.2 NFV

On présente le diagramme de classe du modèle NFV implémenté (figure 3.12) puis un schéma qui indique la correspondance des classes de notre diagramme vers celui de NFV (figure 3.13). Le diagramme de classe ci-dessous présente la classe principale du service virtuel qui NSD(Network Service Descriptor). NSD se compose de la classe de description de VNF (VNFD : Virtual Network Function Description) avec ses constituants (constituentVNFD), de VLD

(Virtual Link Description) qui décrit les liens entre les nœuds de la chaine de service et les points de connexion entre les nœuds (VNFD_connection_point). Chaque nœud de la chaine de fonctions service a une politique d'échelonnement (scaling_policy).

La correspondance établie entre le modèle SFC proposé et le modèle NFV, associe la classe principale ServiceFunctionChain avec NSD (Network Service Descriptor), associe la classe VNFNode avec VNFD (Virtual Network Function Description) et avec VLD aussi (Virtual Link Description) et enfin la classe SFPath avec VNFFGD (VNF Forwarding Graph Description).

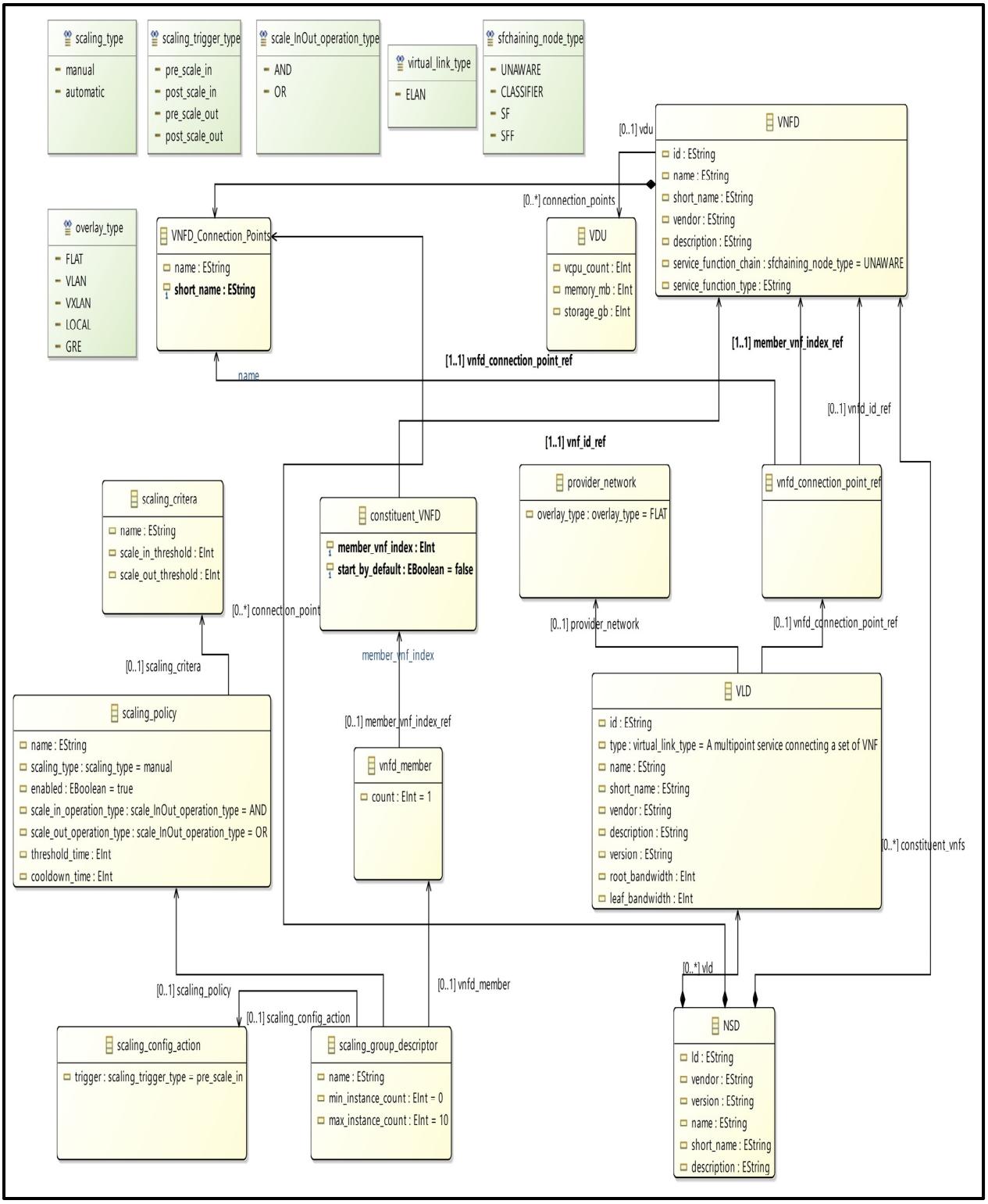


Figure 3.12 Diagramme de classe NFV

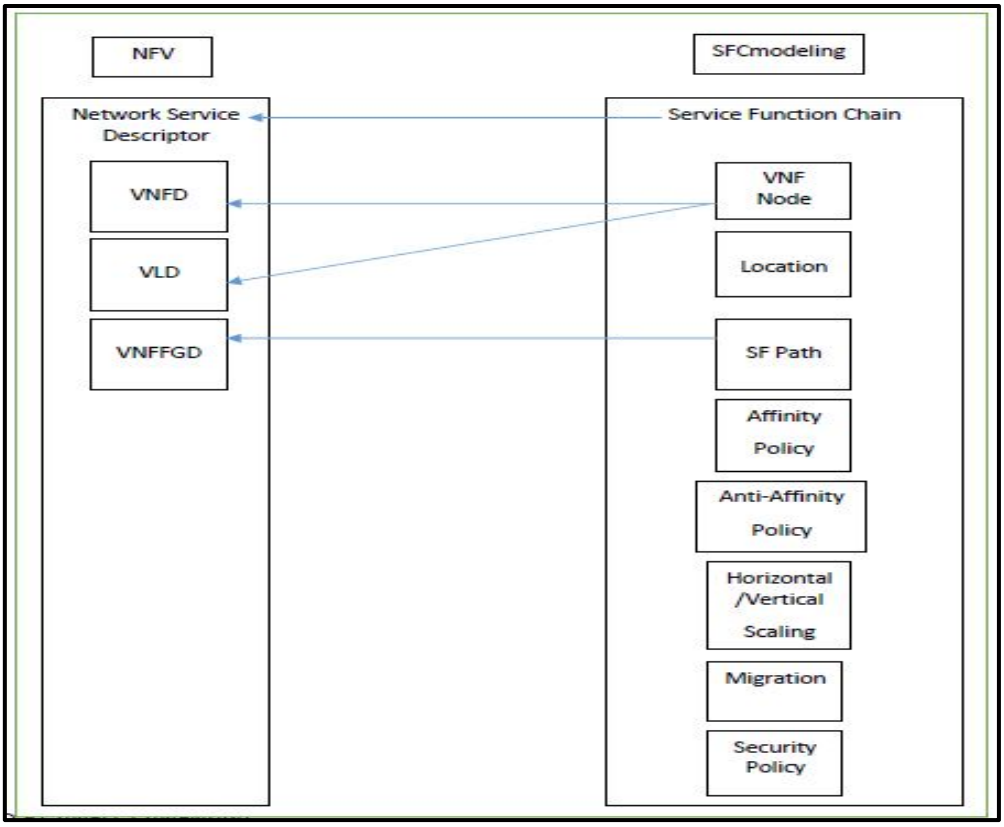


Figure 3.13 Schéma d'échange de données entre le modèle SFC et NFV

3.4.2 Règles de transformation

La règle de transformation est la construction de base dans ATL utilisée pour exprimer la logique de transformation. Les règles ATL peuvent être spécifiées dans un style déclaratif ou dans un style impératif. Les règles ATL déclaratives sont appelées règles correspondantes. Une règle correspondante est composée d'un modèle source et d'un modèle cible. Le modèle de source de règle spécifie un ensemble de types de source (provenant des métamodèles sources et de l'ensemble des types de collection disponibles dans OCL) et une évaluation (en tant qu'expression booléenne dans OCL). Un modèle source est évalué par rapport à un ensemble de correspondances dans les modèles source. Le motif cible est composé d'un ensemble d'éléments. Chacun de ces éléments spécifie un type de cible (à partir du métamodèle cible) et un ensemble de liaisons. Une liaison fait référence à une fonctionnalité du type (c'est-à-dire un

attribut, une référence ou une fin d'association) et spécifie une expression dont la valeur est utilisée pour initialiser la fonctionnalité. (Frédéric Jouault, Ivan Kurtev, 2004). L'extrait de code suivant (figure 3.14) montre la règle de correspondance SFC – TOSCA entre la classe VNFNode et NodeTemplate, qui affecte les propriétés de la classe NodeTemplate aux propriétés de la classe VNFNode :

```
rule VNFNode2PropertyandInterfaceandNodeTemplate {
    from
        s: sfcmodeling!VNFNode
    to
        t: toscamodeling!NodeTemplate (
            type <-  ' '+s.type.toString(),
            metadata <- 'cpus: ' + s.cpuNumber +', memory: '+s.memory+' GB, storage: '+s.storage+' GB',
            description <- ''+ s.name + ' node with ' + s.virtualisationTechnology.toString() +' virtualisation technology'
        ),
        c: toscamodeling!Property (
                name <- 'cpus',
                value <- ''+s.cpuNumber
         ),
         m: toscamodeling!Property (
                name <- 'memory',
                value <- ''+s.memory
         ),
         st: toscamodeling!Property (
                name <- 'storage',
                value <- ''+s.storage
         ),
        i: toscamodeling!Interface (
                type <- 'vport',
                name_property <- 'port_number',
                value_property <- ''+s.portNumber
         )
}
```

Figure 3.14 Règle de correspondance SFC - TOSCA

L'extrait de code suivant (figure 3.15) montre une règle de correspondance SFC – NFV de la classe VNFNode avec VNFD, qui affecte les propriétés de la classe VNFNode aux propriétés de la classe VNFD:

```
rule VNFNode2constituentVNFDandVNFDandVDU {
    from
        s: sfcmodeling!VNFNode
    to
        t: nfvmodeling!constituent_VNFD (
            member_vnf_index <- 1
        ),
        t1: nfvmodeling!VNFD (
            id <- s.id,
            name <- s.name,
            short_name <- s.name,
            vendor <- s.vendor,
            description <- s.name,
            service_function_type <- s.type.toString()
        ),
        t2: nfvmodeling!VDU (
        vcpu_count <- s.cpuNumber,
        memory_mb <- s.memory,
        storage_gb <- s.storage

        ),
        t3: nfvmodeling!VLD (
        id <- s.id,
        name <- s.name,
        short_name <- s.name,
        vendor <- s.vendor,
        description <- s.name,
        version <- '1'
        ),
        t4: nfvmodeling!VNFD_Connection_Points (
        name <- s.name +' vport ' + s.portNumber,
        short_name <- '' + s.cpuNumber

        )
}
```

Figure 3.15 Règle de correspondance SFC – NFV

3.4.3 Outils de développement

Eclipse IDE : c'est un environnement de développement intégré (programme) utilisé par de nombreux programmeurs Java professionnels (ainsi que par des programmeurs utilisant d'autres langages de programmation). C'est un produit Open Source, qui fournit un éditeur permettant d'organiser le travail en plusieurs fichiers qui forment ensemble un projet et dispose d'un compilateur pour pouvoir exécuter les programmes. Plusieurs projets forment un espace de travail. (Leena Razzaq, Meredith Bittrich, 2020).

ATL Development Tools : développé dans le cadre du sous-projet ATL Eclipse / GMT. ATL est accompagné d'un ensemble d'outils construits sur la plate-forme Eclipse. Le moteur ATL est chargé de traiter les tâches ATL principales: compilation et exécution. Les transformations ATL sont compilées en programmes en code octet spécialisé. Le code d'octet est exécuté par la machine virtuelle ATL (VM). La VM est spécialisée dans la gestion des modèles et fournit un ensemble d'instructions pour la manipulation des modèles. (Freddy Allilaire, Jean Bézivin, Frédéric Jouault, Ivan Kurtev, 2006)

3.5 Conclusion

Dans ce chapitre nous avons présenté la conception du modèle générique ainsi que l'implémentation de l'éditeur et de l'outil de transformation vers les modèles TOSCA et NFV.

Dans le prochain chapitre nous présentons l'approche de validation de ce modèle ainsi que la validation de l'éditeur et l'outil de transformation déjà mentionné.

CHAPITRE 4

TEST ET VALIDATION DU MODÈLE SFC ET DES APPROCHES DE TRANSFORMATIONS

4.1 Introduction

Ce quatrième chapitre et dernier, vient tester le travail du mémoire élaboré, par le biais de quelques scénarios de test des chaines de fonctions service qu'on déploie dans le domaine de la virtualisation et l'infonuagique. On formule tout d'abord les scénarios des SFC qui seront utiles pour tester l'éditeur et le moteur de transformation du modèle proposé, à travers des figures contenant les graphes des SFC. Divisé en deux parties, le test consiste à appliquer en premier lieu les scénarios sur le plan de travail de l'éditeur des chaines de fonctions service et puis sur les transformations SFC vers TOSCA et SFC vers NFV. Finalement on termine notre test par les conditions prédéfinis sur l'éditeur.

4.2 Scénarios de test

Les scénarios choisis (figure 4.1 , figure 4.2, figure 4.3, figure 4.4) sont tirés de (Ali Hmaity, 2017). On a également pris les valeurs des ressources des fonctions réseau virtuelles (2 CPU, RAM à 10 Gbytes, stockage à 50 Gbytes, numéro de port entre 1 et 49251).
Les fonctions virtuelles utilisées dans les quatre scénarios sont : NAT (Network Address Translator), FW (Firewall), TM (Traffic Monitor), WOC (WAN Optimization Controller), IDPS (Intrusion Detection and Prevention System) et VOC (Video Optimization Controller).

1. Scénario 1: service web,

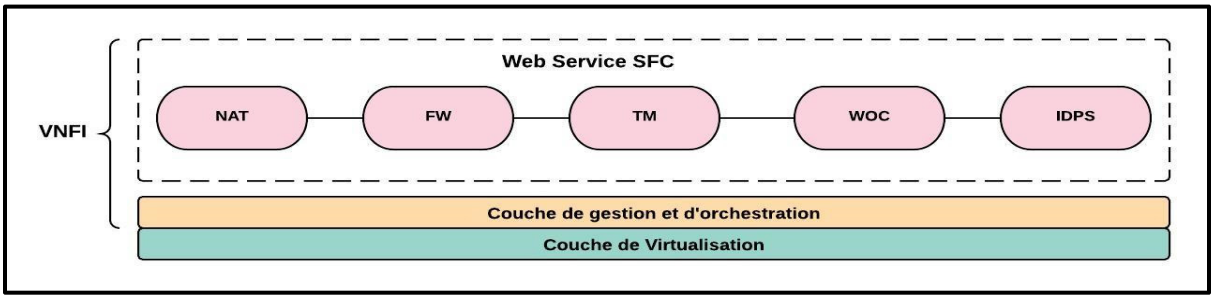


Figure 4.1 SFC 'Web Service' Inspirée de Ali Hmaity (2017)

2. Scénario 2: VoIP,

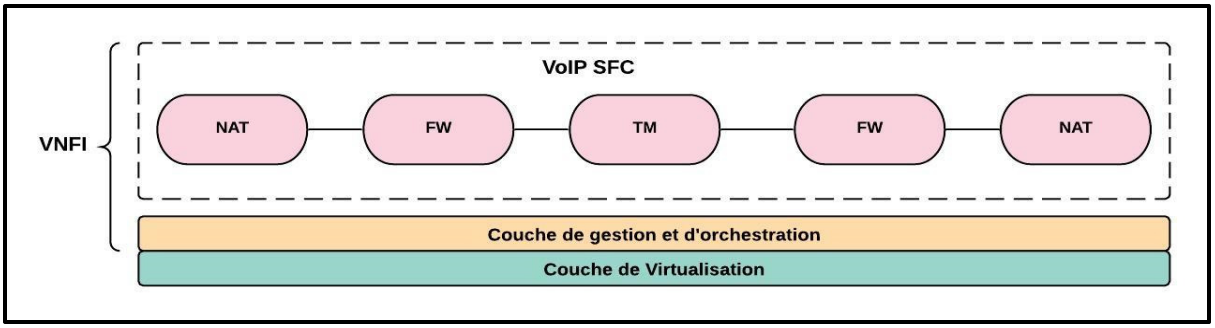


Figure 4.2 SFC 'VoIP'
Inspirée de Ali Hmaity (2017)

3. Scénario 3: video streaming,

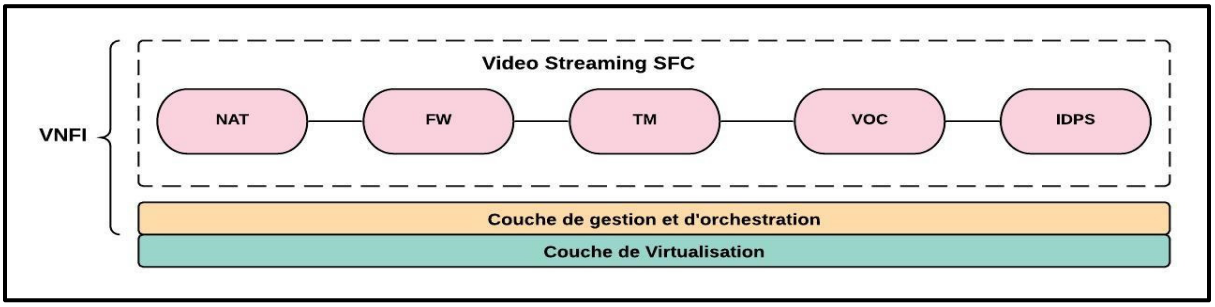


Figure 4.3 SFC 'Video Streaming'
Inspirée de Ali Hmaity (2017)

4. Scénario 4: online gaming.

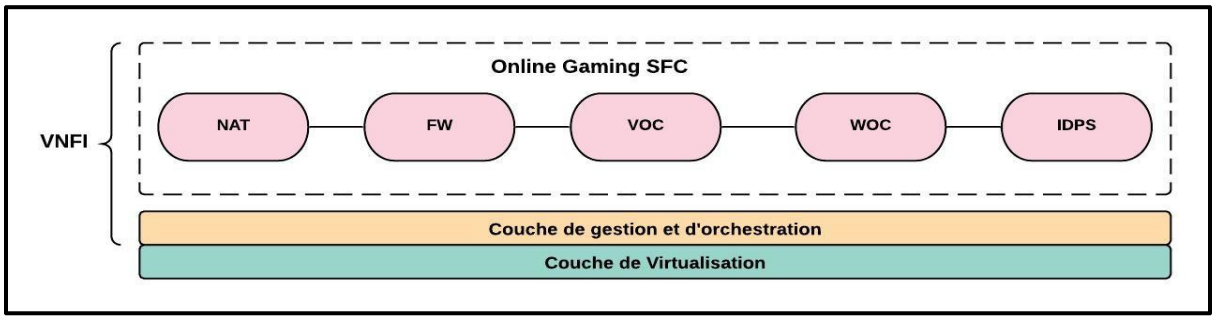


Figure 4.4 SFC ‘online gaming’ Inspirée de Ali Hmaity (2017)

4.3 Test et validation du modèle SFC

La validation du modèle SFC se fait d’une part syntaxiquement en validant les différents éléments du diagramme de classe sur le Framework EMF (Eclipse Modeling Framework) : Classes, attributs, associations, type des attributs, énumérations, etc. Cette validation se fait en traçant le diagramme de classe sur l’environnement Obeo permettant en premier lieu de vérifier les différents composants du diagramme. D’une autre part, la validation se fait d’une manière sémantique une fois la syntaxe est validée, en arrivant à générer le code Java sans problème du fichier. ecore qui contient le diagramme de classe. Cette deuxième validation se fait également par l’environnement Obeo qui génère le code Java automatiquement. Le test du modèle SFC consiste à tester l’éditeur sur lequel s’y fonde, entre autres, pouvant tracer les scénarios de test sur l’éditeur sans problème.

4.3.1 Test du modèle SFC

Ci-dessous (figure 4.5) l'illustration du résultat de validation syntaxique de notre diagramme de classe SFC indiquant qu’aucune erreur n’est détectée. Ce qui fait, on pourra passer à l'étape de validation suivante.

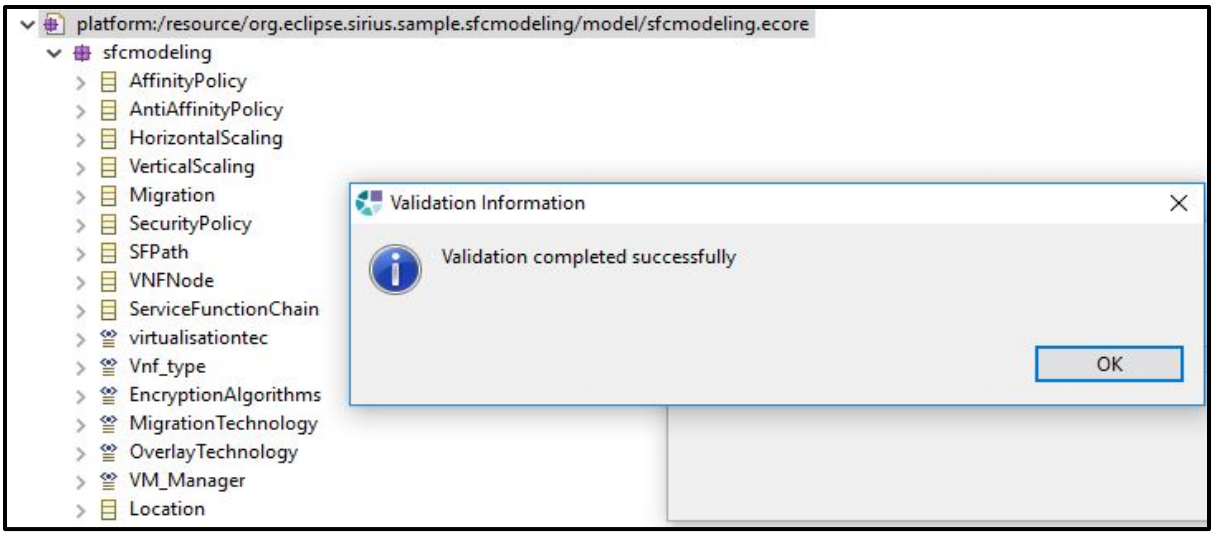


Figure 4.5 Message de validation syntaxique du diagramme SFC

Après la validation syntaxique, une validation sémantique s'est appliquée à notre diagramme et qui a permis la génération du code Java. Ci-dessous (figure 4.6) un extrait du code source Java généré :

```
public void setBandwidth(int newBandwidth) {
    int oldBandwidth = bandwidth;
    bandwidth = newBandwidth;
    if (eNotificationRequired())
        eNotify(new ENotificationImpl(this, Notification.SET, SfcmodelingPackage.SERVICE_FUNCTION_CHAIN__BANDWIDTH,
                oldBandwidth, bandwidth));
}

 * <!-- begin-user-doc -->
public int getAvailability() {

 * <!-- begin-user-doc -->
public void setAvailability(int newAvailability) {
    int oldAvailability = availability;
    availability = newAvailability;
    if (eNotificationRequired())
        eNotify(new ENotificationImpl(this, Notification.SET,
                SfcmodelingPackage.SERVICE_FUNCTION_CHAIN__AVAILABILITY, oldAvailability, availability));
}

 * <!-- begin-user-doc -->
public EList<Location> getLocation() {
    if (location == null) {
        location = new EObjectContainmentEList<Location>(Location.class, this,
                SfcmodelingPackage.SERVICE_FUNCTION_CHAIN__LOCATION);
    }
    return location;
}
```

Figure 4.6 Extrait du code Java

4.3.2 Test de l'éditeur SFC par les scénarios de test

La création des scénarios des tests précédemment décrits a été validée par l'éditeur ainsi que les valeurs des propriétés des composants de la chaîne de fonction service et des règles déjà définies.

Ci-dessous (figure 4.7) une illustration des scénarios de test validés par l'éditeur.

1. Web service,

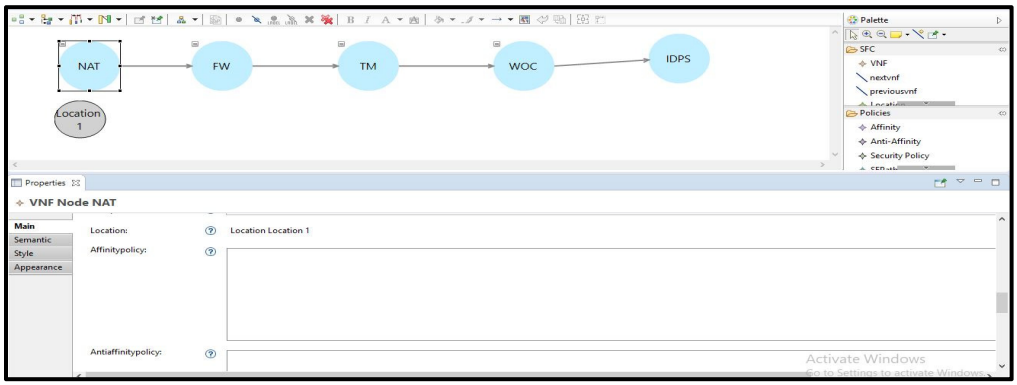


Figure 4.7 Web service

Ci-dessous (figure 4.8) le SFC web service et 'location' :

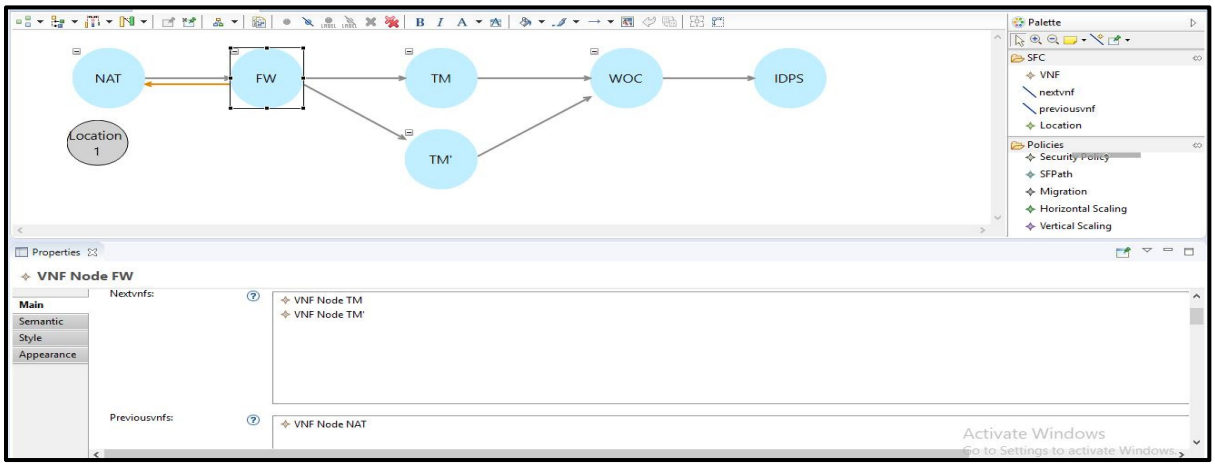


Figure 4.8 Web service et 'location'

Et ci-dessous (figure 4.9) le SFC web service et ‘anti-affinity’ :

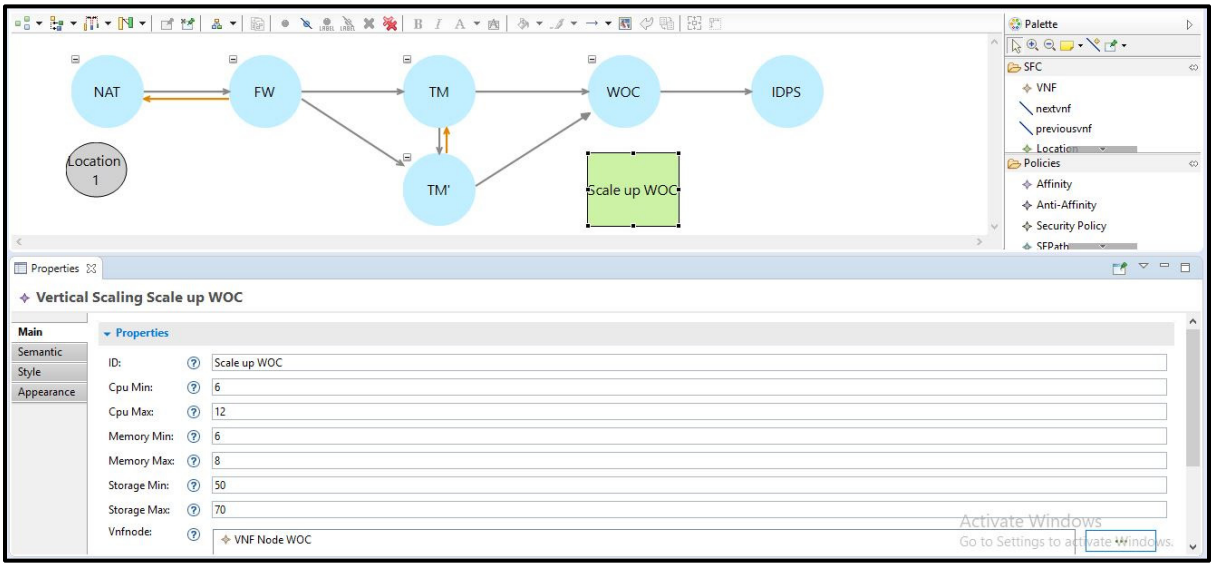


Figure 4.9 Web service et ‘anti-affinity’

2. VoIP,

Ci-dessous (figure 4.10) le SFC VoIP et ‘affinity’ :

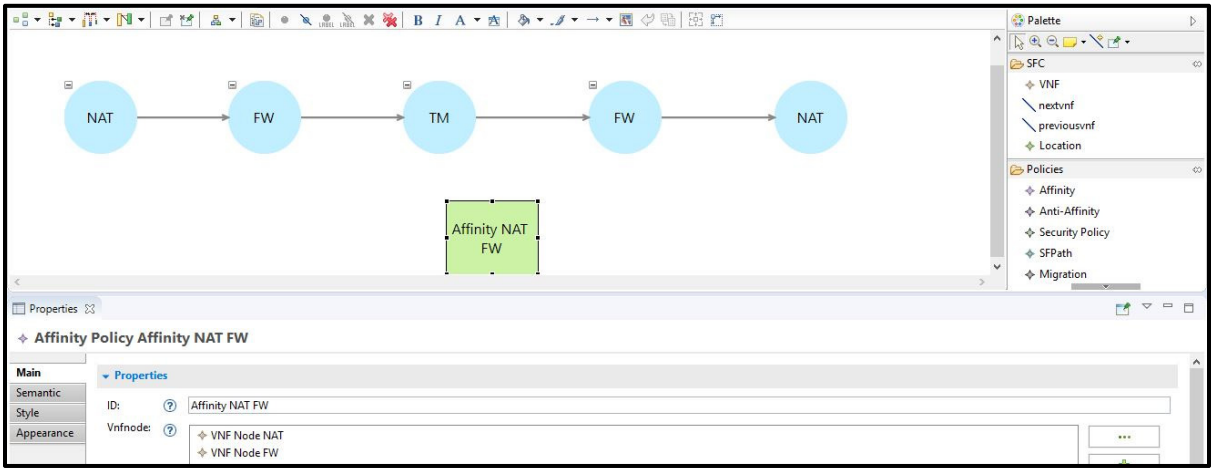


Figure 4.10 VoIP et ‘affinity’

3. Video streaming,

Ci-dessous la SFC ‘video streaming’ (figure 4.11) :

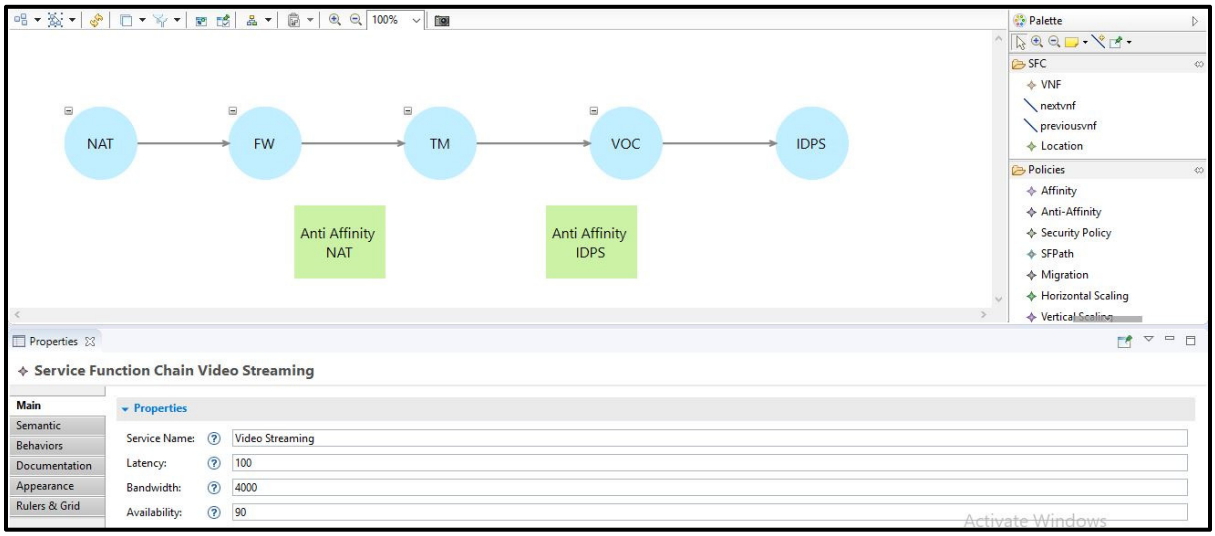


Figure 4.11 Video streaming

4. Online gaming.

Ci-dessous (figure 4.12) la SFC ‘online gaming’ :

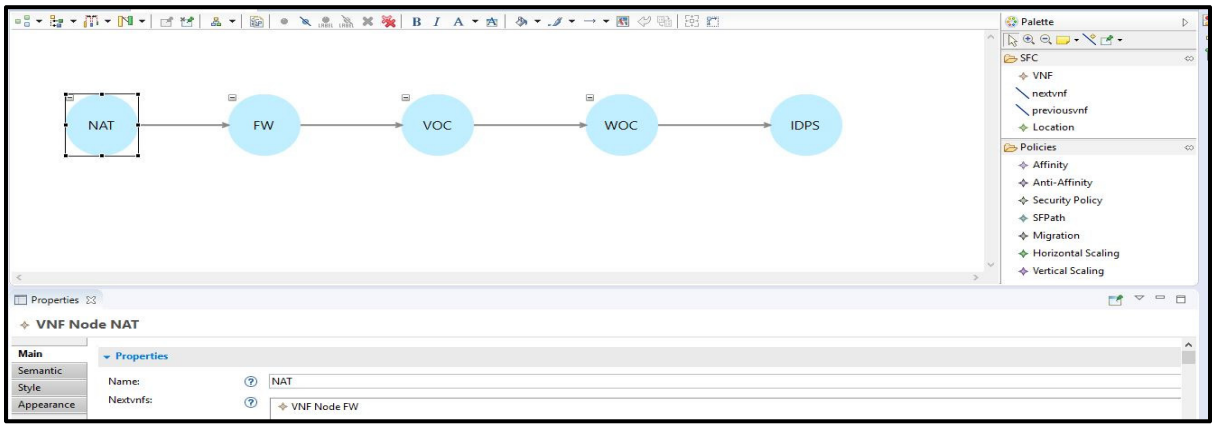


Figure 4.12 ‘Online gaming’

Et ci-dessous (figure 4.13) le SFC ‘online gaming’ contenant le chemin de la SFC saisi dans l’entité SFPath1 :

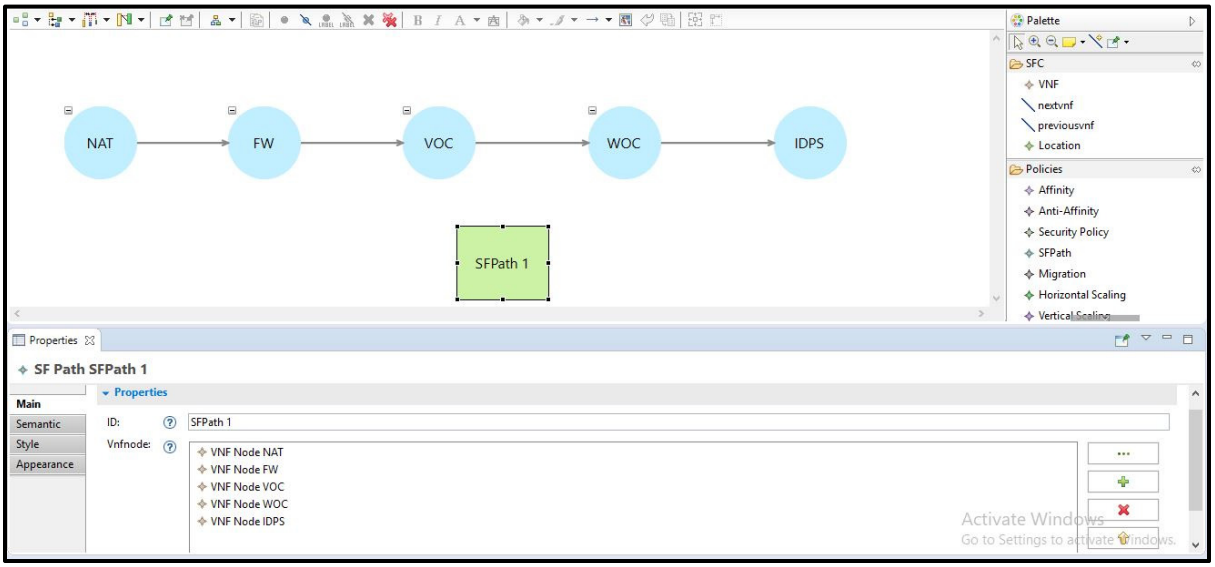


Figure 4.13 ‘Online gaming’ et chemin de la SFC

Ci-dessous (figure 4.14) une illustration des règles appliquées sur les scénarios de test validés par l’éditeur qui a été implémenté par le langage des requêtes AQL (Acceleo Query Language) et validé par l’environnement Obeo:

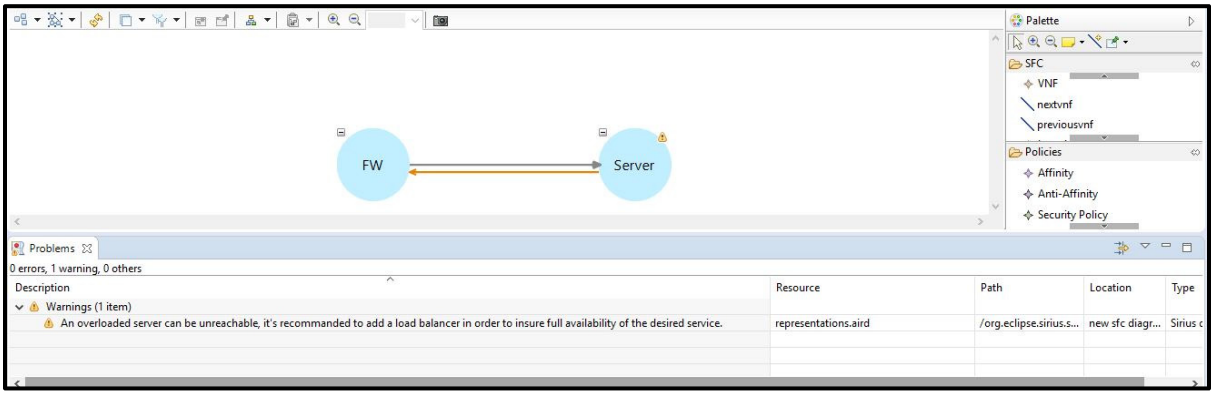


Figure 4.14 Test de règle du load balancer

Ci-dessous (figure 4.15 et figure 4.16) le résultat des règles de vérification des valeurs entières et du lien cyclique.

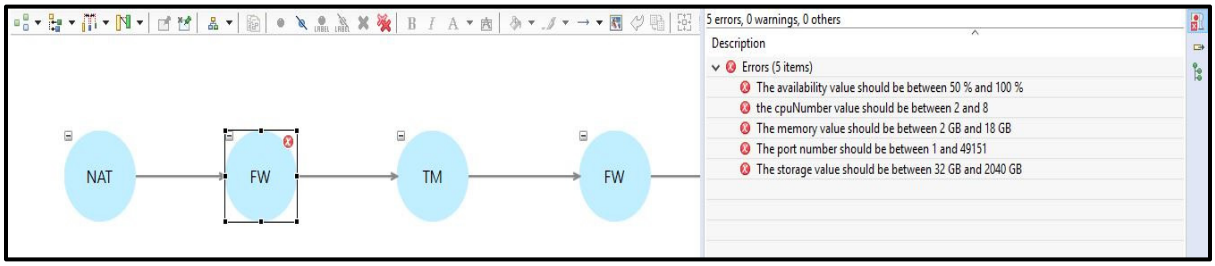


Figure 4.15 Test de règle des valeurs entières

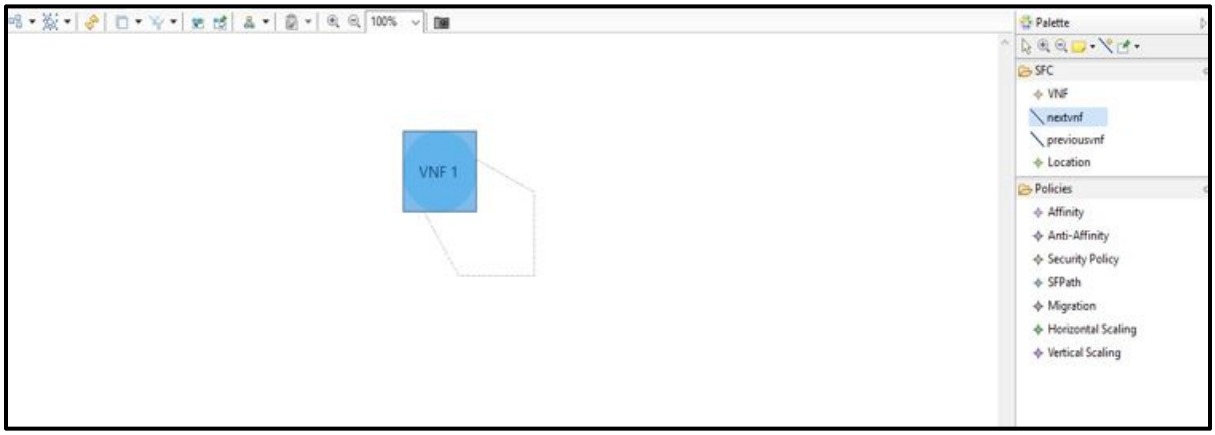


Figure 4.16 Test de règle du lien cyclique

4.4 Test et validation de Transformation du modèle SFC

Afin de tester le modèle SFC, on a développé les règles de transformation ATL pour assurer l'échange des données entre le modèle source et les modèles cibles. Les images ci-dessous (figure 4.17, figure 4.18, figure 4.19 et figure 7.20) présentent les fichiers XMI des scénarios de test avec le modèle source SFC qui seront les entrées de la transformation.

webservice-sfcmodeling.xmi

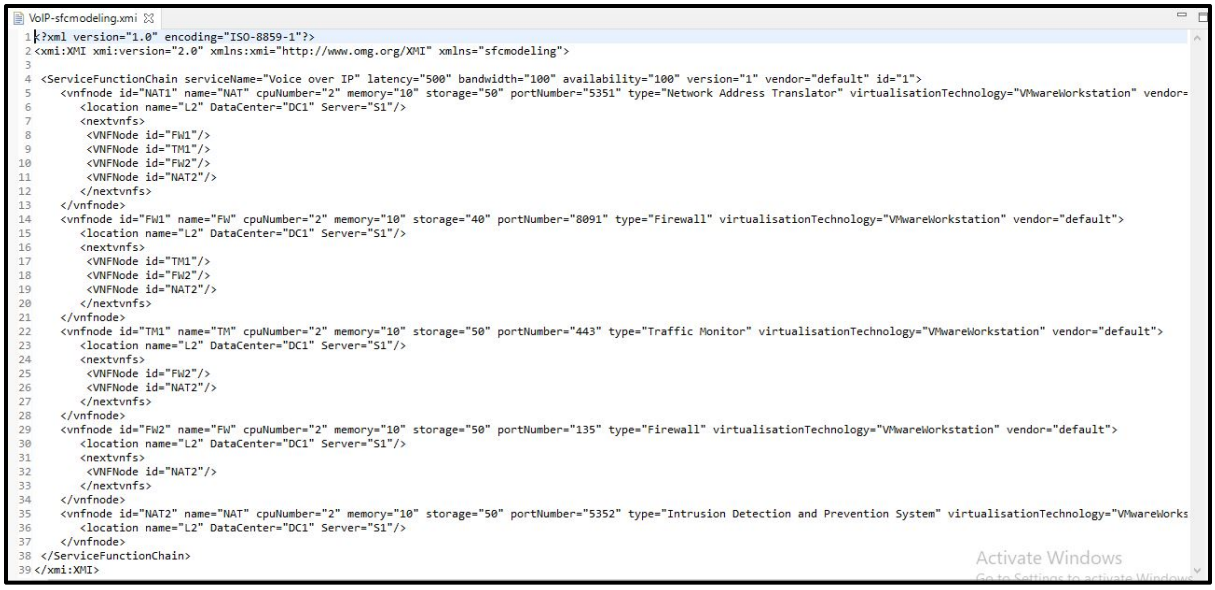
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns="sfcmodeling">

 <ServiceFunctionChain serviceName="web service" latency="500" bandwidth="100" availability="100" version="1" vendor="default" id="1">
    <vnfnode id="NAT1" name="NAT" cpuNumber="2" memory="10" storage="50" portNumber="5351" type="Network Address Translator" virtualisationTechnolo
       <location name="L1" DataCenter="DC1" Server="S1"/>
       <nextvnfs>
        <VNFNode id="FW1"/>
        <VNFNode id="TM1"/>
        <VNFNode id="WOC1"/>
        <VNFNode id="IDPS1"/>
       </nextvnfs>
    </vnfnode>
    <vnfnode id="FW1" name="FW" cpuNumber="2" memory="10" storage="40" portNumber="8091" type="Firewall" virtualisationTechnology="VMwareWorkstatio
       <location name="L1" DataCenter="DC1" Server="S1"/>
       <nextvnfs>
        <VNFNode id="TM1"/>
        <VNFNode id="WOC1"/>
        <VNFNode id="IDPS1"/>
       </nextvnfs>
       <previousvnfs>
        <VNFNode id="NAT1"/>
       </previousvnfs>
    </vnfnode>
    <vnfnode id="TM1" name="TM" cpuNumber="2" memory="10" storage="50" portNumber="443" type="Traffic Monitor" virtualisationTechnology="VMwareWork
       <location name="L1" DataCenter="DC1" Server="S1"/>
       <nextvnfs>
        <VNFNode id="WOC1"/>
        <VNFNode id="IDPS1"/>
       </nextvnfs>
       <previousvnfs>
        <VNFNode id="NAT1"/>
        <VNFNode id="FW1"/>
       </previousvnfs>
    </vnfnode>
    <vnfnode id="WOC1" name="Wan optimization controller" cpuNumber="2" memory="10" storage="50" portNumber="135" type="WAN Optimization Controller
       <location name="L1" DataCenter="DC1" Server="S1"/>
       <nextvnfs>
        <VNFNode id="IDPS1"/>
```


Figure 4.17 Fichier XMI ‘Web Service’ via le modèle SFC source

```
    <vnfnode id="WOC1" name="Wan optimization controller" cpuNumber="2" memory="10" storage="50" portNumber="135" type="WAN Optimization Controller
       <location name="L1" DataCenter="DC1" Server="S1"/>
       <nextvnfs>
        <VNFNode id="IDPS1"/>
       </nextvnfs>
       <previousvnfs>
        <VNFNode id="NAT1"/>
        <VNFNode id="FW1"/>
        <VNFNode id="TM1"/>
       </previousvnfs>
    </vnfnode>
    <vnfnode id="IDPS1" name="IDPS" cpuNumber="2" memory="10" storage="50" portNumber="445" type="Intrusion Detection and Prevention System" virtua
       <location name="L1" DataCenter="DC1" Server="S1"/>
       <previousvnfs>
        <VNFNode id="NAT1"/>
        <VNFNode id="FW1"/>
        <VNFNode id="TM1"/>
        <VNFNode id="WOC1"/>
       </previousvnfs>
    </vnfnode>
    <affinitypolicy ID="AFF1">
        <vnfnode>
            <VNFNode id="NAT1"/>
            <VNFNode id="FW1"/>
        </vnfnode>
    </affinitypolicy>
 </ServiceFunctionChain>
</xmi:XMI>
```


Figure 4.18 Suite du fichier XMI ‘Web Service’ via le modèle SFC source



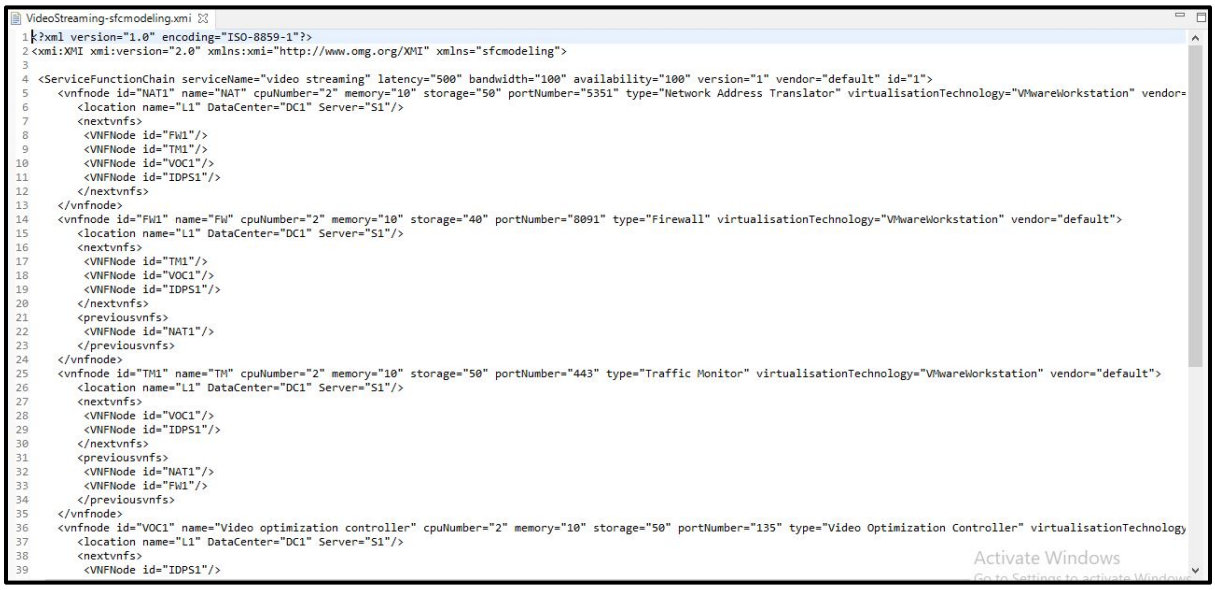
VoIP-sfcmodeling.xmi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns="sfcmodeling">

<ServiceFunctionChain serviceName="Voice over IP" latency="500" bandwidth="100" availability="100" version="1" vendor="default" id="1">
    <vnfnode id="NAT1" name="NAT" cpuNumber="2" memory="10" storage="50" portNumber="5351" type="Network Address Translator" virtualisationTechnology="VMwareWorkstation" vendor=
       <location name="L2" DataCenter="DC1" Server="S1"/>
       <nextvnfs>
        <VNFNode id="FW1"/>
        <VNFNode id="TM1"/>
        <VNFNode id="FW2"/>
        <VNFNode id="NAT2"/>
       </nextvnfs>
    </vnfnode>
    <vnfnode id="FW1" name="FW" cpuNumber="2" memory="10" storage="40" portNumber="8091" type="Firewall" virtualisationTechnology="VMwareWorkstation" vendor="default">
       <location name="L2" DataCenter="DC1" Server="S1"/>
       <nextvnfs>
        <VNFNode id="TM1"/>
        <VNFNode id="FW2"/>
        <VNFNode id="NAT2"/>
       </nextvnfs>
    </vnfnode>
    <vnfnode id="TM1" name="TM" cpuNumber="2" memory="10" storage="50" portNumber="443" type="Traffic Monitor" virtualisationTechnology="VMwareWorkstation" vendor="default">
       <location name="L2" DataCenter="DC1" Server="S1"/>
       <nextvnfs>
        <VNFNode id="FW2"/>
        <VNFNode id="NAT2"/>
       </nextvnfs>
    </vnfnode>
    <vnfnode id="FW2" name="FW" cpuNumber="2" memory="10" storage="50" portNumber="135" type="Firewall" virtualisationTechnology="VMwareWorkstation" vendor="default">
       <location name="L2" DataCenter="DC1" Server="S1"/>
       <nextvnfs>
        <VNFNode id="NAT2"/>
       </nextvnfs>
    </vnfnode>
    <vnfnode id="NAT2" name="NAT" cpuNumber="2" memory="10" storage="50" portNumber="5352" type="Intrusion Detection and Prevention System" virtualisationTechnology="VMwareWorks
       <location name="L2" DataCenter="DC1" Server="S1"/>
    </vnfnode>
</ServiceFunctionChain>
</xmi:XMI>
```

Activate Windows

Figure 4.19 Fichier XMI ‘VoIP’ via le modèle SFC source



VideoStreaming-sfcmodeling.xmi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns="sfcmodeling">

<ServiceFunctionChain serviceName="video streaming" latency="500" bandwidth="100" availability="100" version="1" vendor="default" id="1">
    <vnfnode id="NAT1" name="NAT" cpuNumber="2" memory="10" storage="50" portNumber="5351" type="Network Address Translator" virtualisationTechnology="VMwareWorkstation" vendor=
       <location name="L1" DataCenter="DC1" Server="S1"/>
       <nextvnfs>
        <VNFNode id="FW1"/>
        <VNFNode id="TM1"/>
        <VNFNode id="VOC1"/>
        <VNFNode id="IDPS1"/>
       </nextvnfs>
    </vnfnode>
    <vnfnode id="FW1" name="FW" cpuNumber="2" memory="10" storage="40" portNumber="8091" type="Firewall" virtualisationTechnology="VMwareWorkstation" vendor="default">
       <location name="L1" DataCenter="DC1" Server="S1"/>
       <nextvnfs>
        <VNFNode id="TM1"/>
        <VNFNode id="VOC1"/>
        <VNFNode id="IDPS1"/>
       </nextvnfs>
       <previousvnfs>
        <VNFNode id="NAT1"/>
       </previousvnfs>
    </vnfnode>
    <vnfnode id="TM1" name="TM" cpuNumber="2" memory="10" storage="50" portNumber="443" type="Traffic Monitor" virtualisationTechnology="VMwareWorkstation" vendor="default">
       <location name="L1" DataCenter="DC1" Server="S1"/>
       <nextvnfs>
        <VNFNode id="VOC1"/>
        <VNFNode id="IDPS1"/>
       </nextvnfs>
       <previousvnfs>
        <VNFNode id="NAT1"/>
        <VNFNode id="FW1"/>
       </previousvnfs>
    </vnfnode>
    <vnfnode id="VOC1" name="Video optimization controller" cpuNumber="2" memory="10" storage="50" portNumber="135" type="Video Optimization Controller" virtualisationTechnology
       <location name="L1" DataCenter="DC1" Server="S1"/>
       <nextvnfs>
        <VNFNode id="IDPS1"/>
```

Activate Windows

Figure 4.20 Fichier XMI ‘Video Streaming’ via le modèle SFC source

Figure 4.21 Suite du fichier XMI 'Video Streaming' via le modèle SFC source

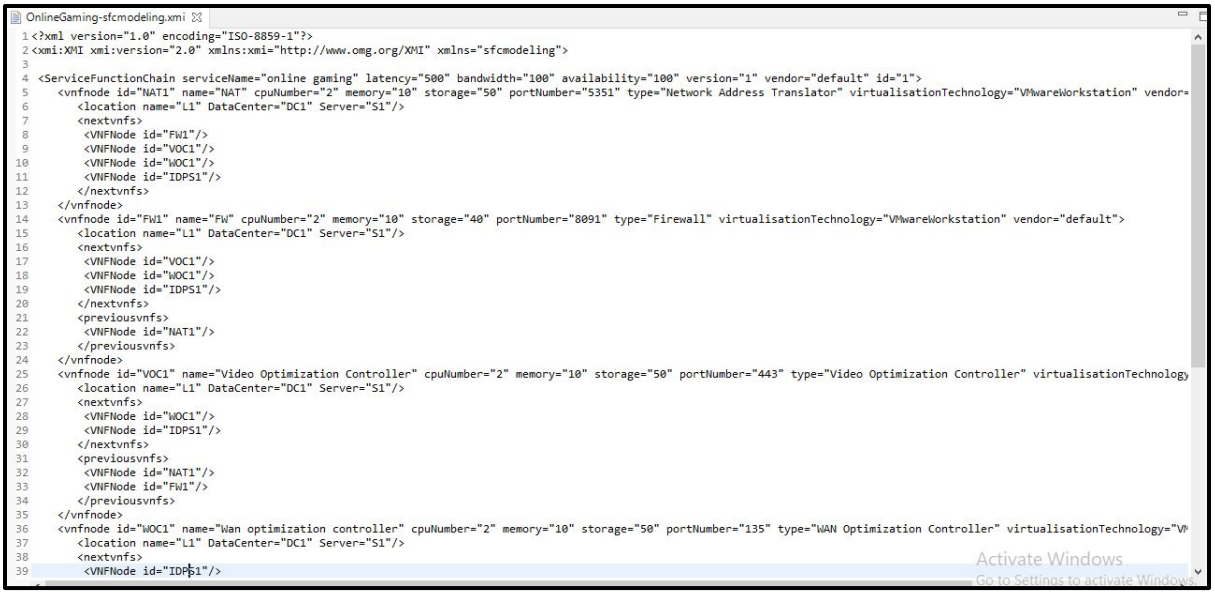


Figure 4.22 Fichier XMI 'Online Gaming' via le modèle SFC source

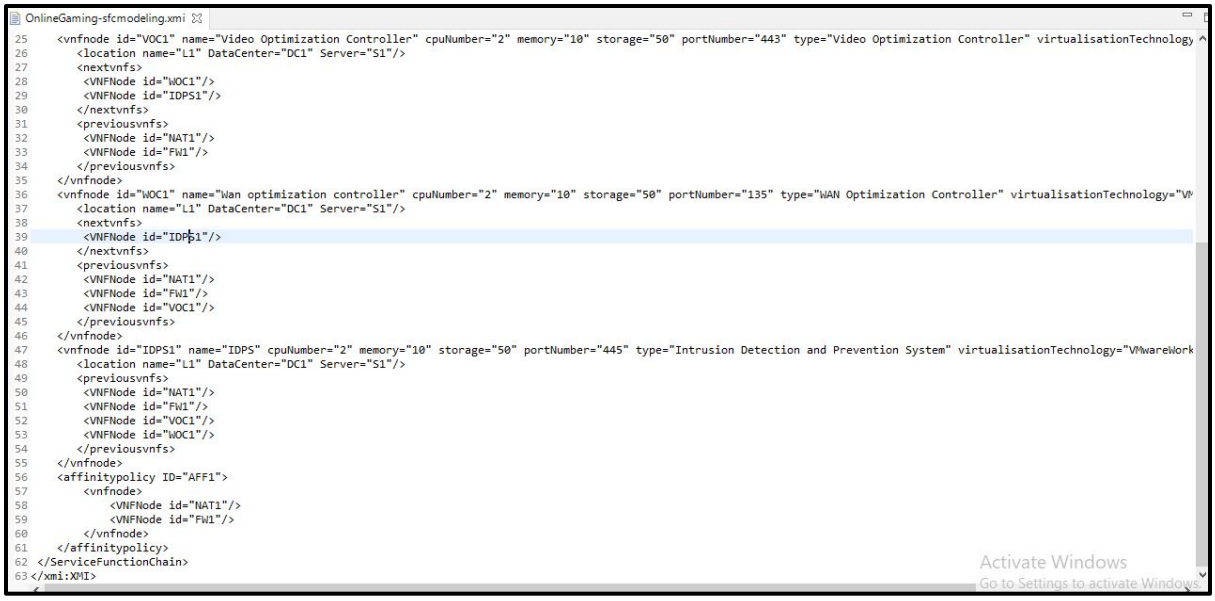


Figure 4.23 Suite du fichier XMI ‘Online Gaming’ via le modèle SFC source

4.4.1 Validation de transformation des scénarios de test vers TOSCA

Premièrement, on présente le schéma (figure 4.24) décrivant la transformation des scénarios de test entre notre modèle de données SFC et celui de TOSCA.

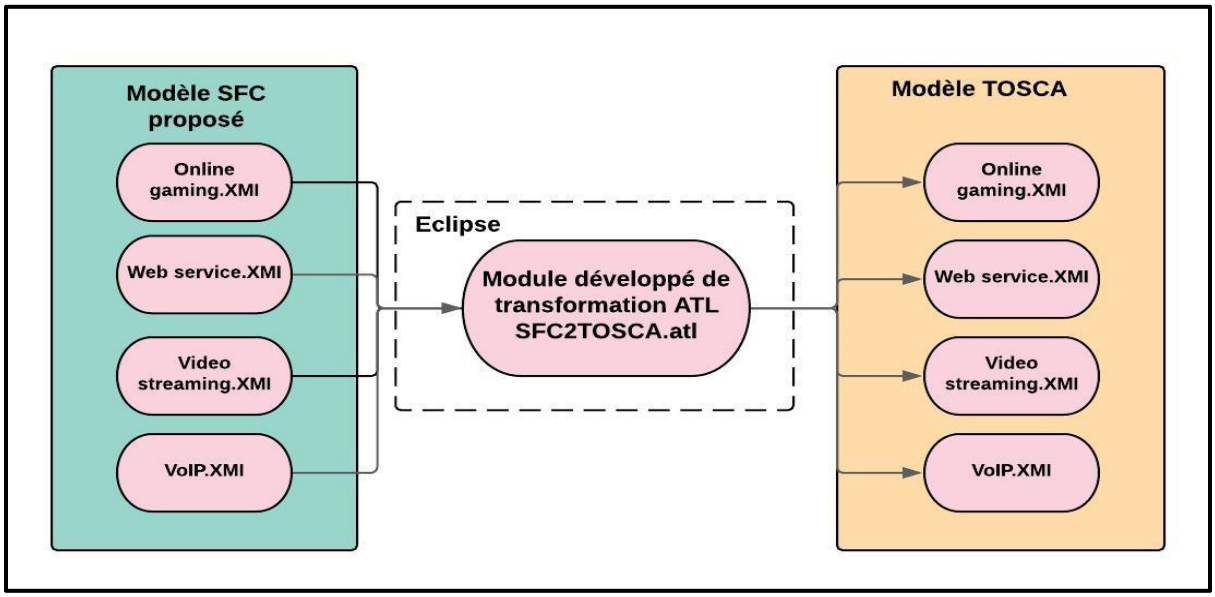
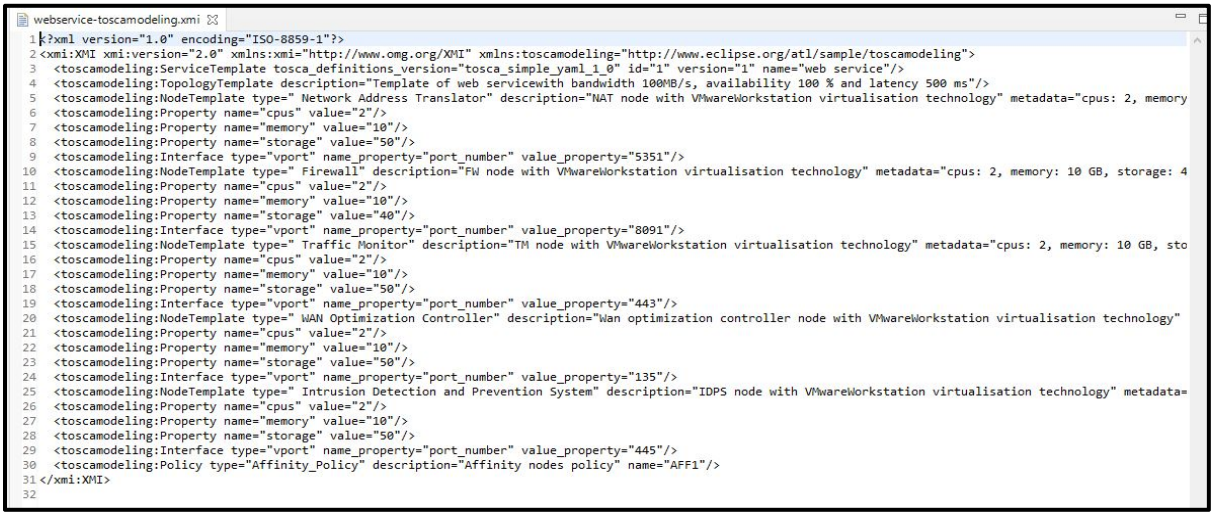


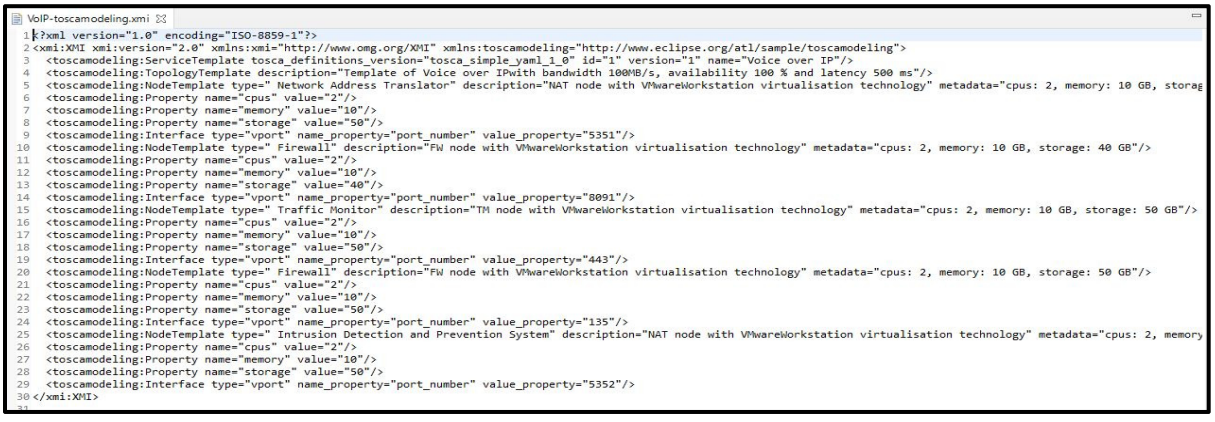
Figure 4.24 Schéma de transformation des scénarios entre SFC et TOSCA

Ce qui suit présente l'application des scénarios précédemment décrits. L'échange des données entre le modèle source et cible se fait en XMI. Les figures ci-dessous (figure 4.25, figure 4.26, figure 4.27 et figure 4.28) présentent les fichiers XMI générés des chaines de fonctions service des scénarios en modèle TOSCA cible lors du l'exécution de la transformation. En effet, les données des scénarios fournies en XMI avec le modèle SFC source se transforment vers le modèle de données TOSCA. Les fichiers résultats de la transformation des quatre scénarios de test respectent le standard TOSCA. Ceci mène à conclure que le modèle SFC proposé est adaptable avec le modèle TOSCA cible.


webservice-toscamodeling.xmi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:toscamodeling="http://www.eclipse.org/atl/sample/toscamodeling">
  <toscamodeling:ServiceTemplate tosca_definitions_version="tosca_simple_yaml_1_0" id="1" version="1" name="web service"/>
  <toscamodeling:TopologyTemplate description="Template of web servicewith bandwidth 100MB/s, availability 100 % and latency 500 ms"/>
  <toscamodeling:NodeTemplate type=" Network Address Translator" description="NAT node with VMwareWorkstation virtualisation technology" metadata="cpus: 2, memory
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="50"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="5351"/>
  <toscamodeling:NodeTemplate type=" Firewall" description="FW node with VMwareWorkstation virtualisation technology" metadata="cpus: 2, memory: 10 GB, storage: 4
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="40"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="8091"/>
  <toscamodeling:NodeTemplate type=" Traffic Monitor" description="TM node with VMwareWorkstation virtualisation technology" metadata="cpus: 2, memory: 10 GB, sto
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="50"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="443"/>
  <toscamodeling:NodeTemplate type=" WAN Optimization Controller" description="Wan optimization controller node with VMwareWorkstation virtualisation technology"
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="50"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="135"/>
  <toscamodeling:NodeTemplate type=" Intrusion Detection and Prevention System" description="IDPS node with VMwareWorkstation virtualisation technology" metadata=
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="50"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="445"/>
  <toscamodeling:Policy type="Affinity_Policy" description="Affinity nodes policy" name="AFF1"/>
</xmi:XMI>
```

Figure 4.25 Fichier XMI 'Web Service' via le modèle TOSCA cible


VoIP-toscamodeling.xmi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:toscamodeling="http://www.eclipse.org/atl/sample/toscamodeling">
  <toscamodeling:ServiceTemplate tosca_definitions_version="tosca_simple_yaml_1_0" id="1" version="1" name="Voice over IP"/>
  <toscamodeling:TopologyTemplate description="Template of Voice over IPwith bandwidth 100MB/s, availability 100 % and latency 500 ms"/>
  <toscamodeling:NodeTemplate type=" Network Address Translator" description="NAT node with VMwareWorkstation virtualisation technology" metadata="cpus: 2, memory: 10 GB, storag
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="50"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="5351"/>
  <toscamodeling:NodeTemplate type=" Firewall" description="FW node with VMwareWorkstation virtualisation technology" metadata="cpus: 2, memory: 10 GB, storage: 40 GB"/>
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="40"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="8091"/>
  <toscamodeling:NodeTemplate type=" Traffic Monitor" description="TM node with VMwareWorkstation virtualisation technology" metadata="cpus: 2, memory: 10 GB, storage: 50 GB"/>
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="50"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="443"/>
  <toscamodeling:NodeTemplate type=" Firewall" description="FW node with VMwareWorkstation virtualisation technology" metadata="cpus: 2, memory: 10 GB, storage: 50 GB"/>
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="50"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="135"/>
  <toscamodeling:NodeTemplate type=" Intrusion Detection and Prevention System" description="NAT node with VMwareWorkstation virtualisation technology" metadata="cpus: 2, memory
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="50"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="5352"/>
</xmi:XMI>
```

Figure 4.26 Fichier XMI 'VoIP' via le modèle TOSCA cible

VideoStreaming-toscamodeling.xmi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:toscamodeling="http://www.eclipse.org/atl/sample/toscamodeling">
  <toscamodeling:ServiceTemplate tosca_definitions_version="tosca_simple_yaml_1_0" id="1" version="1" name="video streaming"/>
  <toscamodeling:TopologyTemplate description="Template of video streamingwith bandwidth 100MB/s, availability 100 % and latency 500 ms"/>
  <toscamodeling:NodeTemplate type=" Network Address Translator" description="NAT node with VMwareWorkstation virtualisation technology" metadata="cpus: 2, memory: 10 GB, storag
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="50"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="5351"/>
  <toscamodeling:NodeTemplate type=" Firewall" description="FW node with VMwareWorkstation virtualisation technology" metadata="cpus: 2, memory: 10 GB, storage: 40 GB"/>
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="40"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="8091"/>
  <toscamodeling:NodeTemplate type=" Traffic Monitor" description="TM node with VMwareWorkstation virtualisation technology" metadata="cpus: 2, memory: 10 GB, storage: 50 GB"/>
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="50"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="443"/>
  <toscamodeling:NodeTemplate type=" Video Optimization Controller" description="Video optimization controller node with VMwareWorkstation virtualisation technology" metadata="c
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="50"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="135"/>
  <toscamodeling:NodeTemplate type=" Intrusion Detection and Prevention System" description="IDPS node with VMwareWorkstation virtualisation technology" metadata="cpus: 2, memor
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="50"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="445"/>
  <toscamodeling:Policy type="Affinity_Policy" description="Affinity nodes policy" name="AFF1"/>
</xmi:XMI>
```

Figure 4.27 Fichier XMI ‘Video Streaming’ via le modèle TOSCA cible

OnlineGaming-toscamodeling.xmi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:toscamodeling="http://www.eclipse.org/atl/sample/toscamodeling">
  <toscamodeling:ServiceTemplate tosca_definitions_version="tosca_simple_yaml_1_0" id="1" version="1" name="online gaming"/>
  <toscamodeling:TopologyTemplate description="Template of online gamingwith bandwidth 100MB/s, availability 100 % and latency 500 ms"/>
  <toscamodeling:NodeTemplate type=" Network Address Translator" description="NAT node with VMwareWorkstation virtualisation technology" metadata="cpus: 2, memory: 10 GB, storag
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="50"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="5351"/>
  <toscamodeling:NodeTemplate type=" Firewall" description="FW node with VMwareWorkstation virtualisation technology" metadata="cpus: 2, memory: 10 GB, storage: 40 GB"/>
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="40"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="8091"/>
  <toscamodeling:NodeTemplate type=" Video Optimization Controller" description="Video Optimization Controller node with VMwareWorkstation virtualisation technology" metadata="c
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="50"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="443"/>
  <toscamodeling:NodeTemplate type=" WAN Optimization Controller" description="Wan optimization controller node with VMwareWorkstation virtualisation technology" metadata="cpus:
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="50"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="135"/>
  <toscamodeling:NodeTemplate type=" Intrusion Detection and Prevention System" description="IDPS node with VMwareWorkstation virtualisation technology" metadata="cpus: 2, memor
  <toscamodeling:Property name="cpus" value="2"/>
  <toscamodeling:Property name="memory" value="10"/>
  <toscamodeling:Property name="storage" value="50"/>
  <toscamodeling:Interface type="vport" name_property="port_number" value_property="445"/>
  <toscamodeling:Policy type="Affinity_Policy" description="Affinity nodes policy" name="AFF1"/>
</xmi:XMI>
```

Figure 4.28 Fichier XMI ‘Online Gaming’ via le modèle TOSCA cible

4.4.2 Validation de transformation des scénarios de test vers NFV

On présente le schéma (figure 4.29) décrivant la transformation des scénarios de test entre notre modèle de données SFC et celui de NFV.

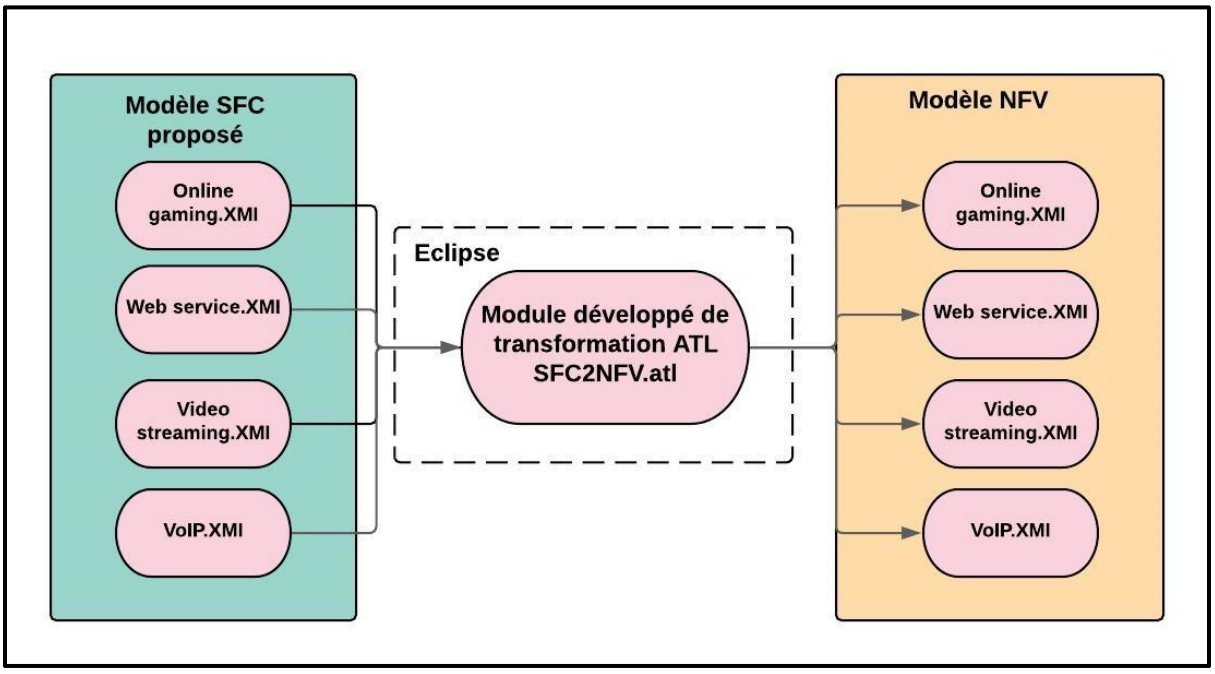


Figure 4.29 Schéma de transformation des scénarios entre SFC et NFV

Ce qui suit présente l'application des scénarios précédemment décrits. L'échange des données entre le modèle source et cible se fait en XMI. Les figures ci-dessous (figure 4.30, figure 4.31, figure 4.32 et figure 4.33) présentent les fichiers XMI générés des chaines de fonctions service des scénarios en modèle NFV cible à l'exécution de la transformation. En effet, les données des scénarios fournies en XMI avec le modèle SFC source se transforment vers le modèle de données NFV. Les fichiers résultats de la transformation des quatre scénarios de test respectent le standard NFV. Ceci mène à conclure que le modèle SFC proposé est adaptable avec le modèle NFV cible.

webservice-nfvmodeling.xmi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:NFVmodeling="http://www.eclipse.org/atl/sample/NFVmodeling">
  <NFVmodeling:NSD Id="1" vendor="default" version="1" name="web service" short_name="web service" description="this is a service for web service ,with latency 500ms and availat
  <NFVmodeling:VLD root_bandwidth="100" leaf_bandwidth="100"/>
  <NFVmodeling:constituent_VNFD member_vnf_index="1"/>
  <NFVmodeling:VNFD id="NAT1" name="NAT" short_name="NAT" vendor="default" description="NAT" service_function_type="Network Address Translator"/>
  <NFVmodeling:VDU vcpu_count="2" memory_mb="10" storage_gb="50"/>
  <NFVmodeling:VLD id="NAT1" name="NAT" short_name="NAT" vendor="default" description="NAT" version="1"/>
  <NFVmodeling:VNFD_Connection_Points name="NAT vport 5351" short_name="2"/>
  <NFVmodeling:constituent_VNFD member_vnf_index="1"/>
  <NFVmodeling:VNFD id="FW1" name="FW" short_name="FW" vendor="default" description="FW" service_function_type="Firewall"/>
  <NFVmodeling:VDU vcpu_count="2" memory_mb="10" storage_gb="40"/>
  <NFVmodeling:VLD id="FW1" name="FW" short_name="FW" vendor="default" description="FW" version="1"/>
  <NFVmodeling:VNFD_Connection_Points name="FW vport 8091" short_name="2"/>
  <NFVmodeling:constituent_VNFD member_vnf_index="1"/>
  <NFVmodeling:VNFD id="TM1" name="TM" short_name="TM" vendor="default" description="TM" service_function_type="Traffic Monitor"/>
  <NFVmodeling:VDU vcpu_count="2" memory_mb="10" storage_gb="50"/>
  <NFVmodeling:VLD id="TM1" name="TM" short_name="TM" vendor="default" description="TM" version="1"/>
  <NFVmodeling:VNFD_Connection_Points name="TM vport 443" short_name="2"/>
  <NFVmodeling:constituent_VNFD member_vnf_index="1"/>
  <NFVmodeling:VNFD id="WOC1" name="Wan optimization controller" short_name="Wan optimization controller" vendor="default" description="Wan optimization controller" service_func
  <NFVmodeling:VDU vcpu_count="2" memory_mb="10" storage_gb="50"/>
  <NFVmodeling:VLD id="WOC1" name="Wan optimization controller" short_name="Wan optimization controller" vendor="default" description="Wan optimization controller" version="1"/>
  <NFVmodeling:VNFD_Connection_Points name="Wan optimization controller vport 135" short_name="2"/>
  <NFVmodeling:constituent_VNFD member_vnf_index="1"/>
  <NFVmodeling:VNFD id="IDPS1" name="IDPS" short_name="IDPS" vendor="default" description="IDPS" service_function_type="Intrusion Detection and Prevention System"/>
  <NFVmodeling:VDU vcpu_count="2" memory_mb="10" storage_gb="50"/>
  <NFVmodeling:VLD id="IDPS1" name="IDPS" short_name="IDPS" vendor="default" description="IDPS" version="1"/>
  <NFVmodeling:VNFD_Connection_Points name="IDPS vport 445" short_name="2"/>
</xmi:XMI>
```

Figure 4.30 Fichier XMI ‘Web Service’ via le modèle NFV cible

VoIP-nfvmodeling.xmi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:NFVmodeling="http://www.eclipse.org/atl/sample/NFVmodeling">
  <NFVmodeling:NSD Id="1" vendor="default" version="1" name="Voice over IP" short_name="Voice over IP" description="this is a service for Voice over IP ,with latency 500ms and a
  <NFVmodeling:VLD root_bandwidth="100" leaf_bandwidth="100"/>
  <NFVmodeling:constituent_VNFD member_vnf_index="1"/>
  <NFVmodeling:VNFD id="NAT1" name="NAT" short_name="NAT" vendor="default" description="NAT" service_function_type="Network Address Translator"/>
  <NFVmodeling:VDU vcpu_count="2" memory_mb="10" storage_gb="50"/>
  <NFVmodeling:VLD id="NAT1" name="NAT" short_name="NAT" vendor="default" description="NAT" version="1"/>
  <NFVmodeling:VNFD_Connection_Points name="NAT vport 5351" short_name="2"/>
  <NFVmodeling:constituent_VNFD member_vnf_index="1"/>
  <NFVmodeling:VNFD id="FW1" name="FW" short_name="FW" vendor="default" description="FW" service_function_type="Firewall"/>
  <NFVmodeling:VDU vcpu_count="2" memory_mb="10" storage_gb="40"/>
  <NFVmodeling:VLD id="FW1" name="FW" short_name="FW" vendor="default" description="FW" version="1"/>
  <NFVmodeling:VNFD_Connection_Points name="FW vport 8091" short_name="2"/>
  <NFVmodeling:constituent_VNFD member_vnf_index="1"/>
  <NFVmodeling:VNFD id="TM1" name="TM" short_name="TM" vendor="default" description="TM" service_function_type="Traffic Monitor"/>
  <NFVmodeling:VDU vcpu_count="2" memory_mb="10" storage_gb="50"/>
  <NFVmodeling:VLD id="TM1" name="TM" short_name="TM" vendor="default" description="TM" version="1"/>
  <NFVmodeling:VNFD_Connection_Points name="TM vport 443" short_name="2"/>
  <NFVmodeling:constituent_VNFD member_vnf_index="1"/>
  <NFVmodeling:VNFD id="FW2" name="FW" short_name="FW" vendor="default" description="FW" service_function_type="Firewall"/>
  <NFVmodeling:VDU vcpu_count="2" memory_mb="10" storage_gb="50"/>
  <NFVmodeling:VLD id="FW2" name="FW" short_name="FW" vendor="default" description="FW" version="1"/>
  <NFVmodeling:VNFD_Connection_Points name="FW vport 135" short_name="2"/>
  <NFVmodeling:constituent_VNFD member_vnf_index="1"/>
  <NFVmodeling:VNFD id="NAT2" name="NAT" short_name="NAT" vendor="default" description="NAT" service_function_type="Intrusion Detection and Prevention System"/>
  <NFVmodeling:VDU vcpu_count="2" memory_mb="10" storage_gb="50"/>
  <NFVmodeling:VLD id="NAT2" name="NAT" short_name="NAT" vendor="default" description="NAT" version="1"/>
  <NFVmodeling:VNFD_Connection_Points name="NAT vport 5352" short_name="2"/>
</xmi:XMI>
```

Figure 4.31 Fichier XMI ‘VoIP’ via le modèle NFV cible

VideoStreaming-nfvmodeling.xmi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:NFVmodeling="http://www.eclipse.org/atl/sample/NFVmodeling">
  <NFVmodeling:NSD Id="1" vendor="default" version="1" name="video streaming" short_name="video streaming" description="this is a service for video streaming ,with latency 500ms
  <NFVmodeling:VLD root_bandwidth="100" leaf_bandwidth="100"/>
  <NFVmodeling:constituent_VNFD member_vnf_index="1"/>
  <NFVmodeling:VNFD id="NAT1" name="NAT" short_name="NAT" vendor="default" description="NAT" service_function_type="Network Address Translator"/>
  <NFVmodeling:VDU vcpu_count="2" memory_mb="10" storage_gb="50"/>
  <NFVmodeling:VLD id="NAT1" name="NAT" short_name="NAT" vendor="default" description="NAT" version="1"/>
  <NFVmodeling:VNFD_Connection_Points name="NAT vport 5351" short_name="2"/>
  <NFVmodeling:constituent_VNFD member_vnf_index="1"/>
  <NFVmodeling:VNFD id="FW1" name="FW" short_name="FW" vendor="default" description="FW" service_function_type="Firewall"/>
  <NFVmodeling:VDU vcpu_count="2" memory_mb="10" storage_gb="40"/>
  <NFVmodeling:VLD id="FW1" name="FW" short_name="FW" vendor="default" description="FW" version="1"/>
  <NFVmodeling:VNFD_Connection_Points name="FW vport 8091" short_name="2"/>
  <NFVmodeling:constituent_VNFD member_vnf_index="1"/>
  <NFVmodeling:VNFD id="TM1" name="TM" short_name="TM" vendor="default" description="TM" service_function_type="Traffic Monitor"/>
  <NFVmodeling:VDU vcpu_count="2" memory_mb="10" storage_gb="50"/>
  <NFVmodeling:VLD id="TM1" name="TM" short_name="TM" vendor="default" description="TM" version="1"/>
  <NFVmodeling:VNFD_Connection_Points name="TM vport 443" short_name="2"/>
  <NFVmodeling:constituent_VNFD member_vnf_index="1"/>
  <NFVmodeling:VNFD id="VOC1" name="Video optimization controller" short_name="Video optimization controller" vendor="default" description="Video optimization controller" servic
  <NFVmodeling:VDU vcpu_count="2" memory_mb="10" storage_gb="50"/>
  <NFVmodeling:VLD id="VOC1" name="Video optimization controller" short_name="Video optimization controller" vendor="default" description="Video optimization controller" versior
  <NFVmodeling:VNFD_Connection_Points name="Video optimization controller vport 135" short_name="2"/>
  <NFVmodeling:constituent_VNFD member_vnf_index="1"/>
  <NFVmodeling:VNFD id="IDPS1" name="IDPS" short_name="IDPS" vendor="default" description="IDPS" service_function_type="Intrusion Detection and Prevention System"/>
  <NFVmodeling:VDU vcpu_count="2" memory_mb="10" storage_gb="50"/>
  <NFVmodeling:VLD id="IDPS1" name="IDPS" short_name="IDPS" vendor="default" description="IDPS" version="1"/>
  <NFVmodeling:VNFD_Connection_Points name="IDPS vport 445" short_name="2"/>
</xmi:XMI>
```

Figure 4.32 Fichier XMI ‘Video Streaming’ via le modèle NFV cible

Figure 4.33 Fichier XMI ‘Online Gaming’ via le modèle NFV cible

4.5 Conclusion et perspectives

Ce chapitre a montré comment valider l’approche de modélisation de chaine de fonctions service ainsi que l’approche de transformation vers les deux modèles de descriptions de chaines de fonctions service TOSCA et NFV. Cette validation a été effectué principalement par le biais des quatre scénarios ‘online-gaming’, ‘web service’, ‘voip’ et ‘video streaming’. La validation de la partie modélisation a été établie en se basant sur :

- en premier lieu, la validation syntaxique et sémantique du modèle générique SFC;
- en deuxième lieu, la validation de ce modèle à travers la création de l’éditeur SFC;
- en Dernier lieu, la validation de l’éditeur SFC en appliquant les quatre scénarios de test.

Quant au choix de l’approche de validation de la partie transformation du modèle générique SFC, elle a été basé sur :

- premièrement, une transformation des scénarios de tests vers le modèle TOSCA;
- deuxièmement, une transformation des scénarios de tests vers le modèle NFV.

4.6 Limites de cette recherche

À la lumière de cette recherche, nous estimons que l'approche proposée et les contributions réalisées permettent l'amélioration de la modélisation des services virtuels. Cependant, le modèle générique SFC ainsi que son éditeur proposé présentent quelques limitations qui sont :

- la limitation sur le plan d'optimisation du chainage des fonctions service, c'est-à-dire que notre éditeur n'est pas encore capable d'appliquer une composition dynamique des nœuds de la chaine du modèle saisi ;
- l'éditeur ne comporte pas des options pour la simulation du trafic réseau ou des ressources permettant l'évaluation de performance du modèle SFC saisi ;
- le moteur de transformation développé support seulement les chaines de fonctions service linéaires.

CONCLUSION GÉNÉRALE

La variété des plateformes de déploiement des services virtuels ainsi que la variété des modèles de description des services virtuels que ces plateformes utilisent et le besoin d'un outil de description des services virtuels par les clients demandeurs et les orchestrateurs était la raison principale du besoin d'un modèle standard abstrait regroupant les fonctionnalités de base et communes pour faciliter la modélisation des chaines de fonctions services. Pour remédier à cette problématique, cette recherche propose un modèle abstrait de chaine de fonctions service et les approches de transformations a adopté permettant leurs adaptations vers les modèles connus. Une investigation a été mené pour répondre aux questions suivantes:

Quelles sont les types, les techniques et les architectures de la virtualisation, les composants d'une chaine de fonction service et les étapes de chainage ?

Dans ce travail de mémoire, nous avons étudié les différents concepts liés à la virtualisation afin de se familiariser avec le domaine. Nous avons également vu le concept de chaine de fonctions service et ses composant ainsi les étapes de chainage ou on s'est focalisé sur l'étape de description des chaines de service. C'est au niveau de cette étape ou notre travail se manifeste.

Quel est l'approche de modélisation la mieux adaptée pour concevoir le modèle proposé et quelle est l'approche de transformation adoptée ?

À la lumière de la revue de littérature faite sur l'approche dirigée par modèles et les approches de transformation existantes, on a pu choisir l'approche de modélisation des SFC se basant sur le Framework EMF ainsi que l'approche de transformation hybride utilisant l'outil ATL.

Comment le modèle de description des chaines de fonctions virtuelles et l'approche de transformation ont été validé ?

Le modèle conçu a été validé sur plusieurs phases :

- la validation syntaxique et sémantique du modèle UML sur le cadre EMF;
- la validation fonctionnelle du modèle à travers l'éditeur SFC implémenté dans cette étude;
- la validation de l'approche de transformation du modèle proposé vers les modèles TOSCA et NFV par le biais de l'outil de transformation implémenté dans le cadre de cette étude via 'ATL Development Tools'.

Les contributions de notre étude sont :

- la proposition des modèles SFC génériques;
- l'implémentation d'un éditeur de chaines de fonctions service;
- l'implémentation de l'outil de transformation M2M du modèle proposé vers les modèles TOSCA et NFV.

La conception d'un modèle générique de modélisation des SFC ainsi que le développement d'un éditeur basé sur ce modèle peuvent avoir un grand impact sur l'industrie. L'avantage d'avoir un tel outil est de permettre une conception générique de chaine de fonctions services sans avoir besoin de connaitre les spécificités des modèles utilisés par les plateformes de déploiements. Aussi le développement de l'outil de transformation proposé dans cette étude peut être très bénéfique en terme de portabilité des architectures virtuelles conçues dans l'éditeur SFC puisqu'il permet la transformation à la demande de l'architecture conçue vers un modèle de données cible choisi (TOSCA et NFV dans notre cas d'étude).

Plusieurs travaux futurs pourraient être élaborés en se basant sur ce mémoire :

- ajouter des méthodes d'optimisation des chaines de fonctions services au niveau de l'éditeur;
- ajouter des méthodes d'évaluation des performances des chaines de fonctions services au niveau de l'éditeur;
- adapter l'outil de transformation pour les chaines de fonctions services non linéaires;
- appliquer les scénarios de sorties sur un environnement (exemple : parseur) de déploiement des services virtuels.

LISTE DE RÉFÉRENCES BILIOGRAPHIQUES

Ali Hmaity. (2017, Decembre), Protection Strategies for Virtual Network Functions Placement and Service Chains Provisioning. *Proceedings of International Workshop on Resilient Networks Design and Modeling*, 373-387.

Abdelali, E, Naoual, B. & Sefiani, N. (2013), Architecture dirigée par les modèles : Méthodes et outils de transformation de modèles. *La cinquième édition des Journées Doctorales en Technologies de l'Information et de la Communication*, p. 7.

Acceleo Query Language Query.(2020). Query and navigate in EMF models. Repéré à https://www.eclipse.org/acceleo/documentation/.

Barros, M. T., Reinaldo, C. G., de Alencar, M. S. et Anderson, F. C. (2013). Feature filtering techniques applied in ip traffic classification. *IADIS International Conference WWW/Internet*, 227-234.

Basili, V. R., et Richard W S. (1991). Paradigms for experimentation and empirical studies in software engineering. *Reliability Engineering & System Safety,* 32(1), 171-191.

Basili, V. R., Richard W S & David H H. (1986). Experimentation in software engineering. *Software Engineering, IEEE Transactions on*, SE-12(7), 733-743.

Bo ,Y., Xingwei, W., Keqin, L., Sajal k.D. & Min H. (2018). A comprehensive survey of Network Function Virtualization. *Computer Networks*,133, 51.

Bari, M., Chowdhury S.R., Ahmed, R. et al . (2015), On orchestrating virtual network functions in NFV. *International Conference on Network and Service Management*, 50-56.

Bo, H. & Vijay G.. (2015, 19 Fevier), Network Function Virtualization: Challenges and Opportunities for Innovations. *IEEE Communications*.53(2), 90-97.

Bhakti, H. R., Rahmadani, P., Deci, I., Marnis, N., Ibnu Rasyid, M. (2020, Novembre), UML Modeling and Black Box Testing Methods in the School Payment Information System. *Journal Mantik*,4(3), 1634-1640.

Cisco Systems. (2015), *Problem Statement for Service Function Chaining*. Norme CS 7498 . Repéré à http://www.rfc-editor.org/rfc/rfc7498.txt

Deval, B., Raj, J., Mohammed, S. & Aiman, E. (2016, Novembre). A Survey on Service Function Chaining. *Journal of Network and Computer Applications*, p 18.

Dong, J & Yang, S. (2006), QVT Based model transformation for design pattern evolutions, *Proceedings of the 10th IASTED international conference on Internet and multimedia systems and applications*, p. 7.

Ericsson. (2015), *Service Function Chaining (SFC) Architecture*. Norme Ericsson 7665 . Stokholm : Suède: Ericsson.

ETSI-developers. (2019). Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on YANG Specification. Norme ETSI GS NFV-SOL 006 . CEPT : ETSI.

Elias, J., Martignon, F., Paris, S. et al. (2017, 5 Novembre), Efficient Orchestration Mechanisms for Congestion Mitigation in NFV: Models and Algorithms. *IEEE Transactions on Services Computing*, 10(4), 534-546.

Frédéric, J., Ivan, K.(2005), Transforming models with ATL. *Proceedings of the Model Transformations in Practice Workshop at MoDELS*, p. 10.

Freddy, A., Jean, B,, Frédéric, J., Ivan, K. (2006), ATL: Eclipse Support for Model Transformation ». Proceedings of the Eclipse Technology eXchange workshop, p. 5.

Herrera, J.G. & Botero, J.F. (2016, 5 Novembre), Resource allocation in NFV:A comprehensive survey. *IEEE Transactions on Network andService Management*, 13(3), 518-532 .

IBM Software. (1998). XML Metadata Interchange. Repéré à http://xml.coverpages.org/xmi-ibm980612.html.

Jean, B., Guillaume, H., Frédéric, J., Ivan, K. and William, Piers. (2004, Decembre), Bridging the MS/DSL Tools and the Eclipse Modeling Framework. ATLAS Group, INRIA & LINA, University of Nantes, p. 7.

Ludovic, M. (2010), Formalisation d'une approche d'Ingénierie Dirigée par les Modèles appliquée au domaine de la Gestion des Données de Référence. (Thèse de doctorat,Université Paris VIII, France). Repéré à http://www.theses.fr/2010PA083184.

Louis, M. R., Nicholas. M., Dimitrios, S. K., & Richard, F. P. (2012), A feature model for model-to-text transformation languages. *Modeling in Software Engineering*, p. 7.

Leena, R. & Meredith, B.(2020). Introduction to Eclipse and Simple Data Definitions. Repéré à https://course.ccs.neu.edu/cs2510/lab1.html

Marouen, M., Imen, G. B.Y. & Zeghlache D.(2016), Agile service manager for 5G. *Network Operations and Management Symposium*, p.7 .

Mirjalily, G. & Luo Z. (2018, July), Optimal Network Function Virtualization and Service Function Chaining: A survey. *Chinese Journal of Electronics*, 704-717.

Obeo designer.(2020). The easiest way to define your own modeling tools. Repéré à https://www.obeodesigner.com/en/product

Porres, I. (2003), Model refactorings as rule-based update transformations. *International Conference on the Unified Modeling Language*, 159-174.

Reuel Nathan Wandji, N. (2018). Placement et chaînage dynamique de fonctions réseau surmultiples centres de données.(mémoire de maitrise, École Polytechnique de Montréal, Montréal). Repéré à https://publications.polymtl.ca/3698/

Stephan, M., Stevenson, A. (2009), A comparative look at model transformation languages, Software Technology Lab, Queen's University, p. 5.

Sirius Obeo designer. (2020). Sirius A graphic model is worth a thousand words. Repéré à https://www.obeodesigner.com/en/product/sirius

Tobias, B., Gerd, B., Frank L. & Thomas S. (2012, May). Portable Cloud Services Using TOSCA. *IEEE Internet Computing*, 80-84.

Van-Ca, N. & Anh-Vu, V. (2017), An Experimental Study of Security for Service Function Chaining. *International Conference on Ubiquitous and Future Networks*, 797-799.

Vladimir, V., Mirjana, M. & Branko, P. (2014),Sirius : A rapid development of DSM Graphical Editor. *18th International conference of Intelligent Engineering Systems*, 233-238.

Wei, Y. & Carol, F. (2016), A Survey on Security in Network Functions Virtualization. *IEEE Conference on Network Softwarization*, 5 .

Yong, L. & Min C. (2015, 16 Décembre). Software-Defined Network Function Virtualization: A Survey. *IEEE ACCESS,* p. 12.

Zhani, M. F. (2018). MTI777: Conception de services réseautiques. École de Technologie Superieure.