Une architecture convergente pour une continuité et personnalisation de services : aspects architectural et fonctionnel

Rachad Nassar

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2012-ENST-025

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Informatique et Réseaux »

présentée et soutenue publiquement par

Rachad NASSAR

le 27 Juin 2012

Une Architecture Convergente pour une Continuité et Personnalisation de Services : Aspects Architectural et Fonctionnel

Directrice de thèse : **Pr. Noëmie SIMONI**

Jury

M. Omar CHERKAOUI, Professeur, Université du Québec à Montréal	Rapporteur
M. Ken CHEN, Professeur, Université Paris 13	Rapporteur
Mme. Francine KRIEF, Professeur, Université de Bordeaux	Examinatrice
M. Elie NAJM, Professeur, Télécom ParisTech	Examinateur
M. Antoine BOUTIGNON, Directeur Réseau et Grands Projets, SFR	Examinateur
Mme. Sylvie VIGNES, Maître de conférence, Télécom ParisTech	Examinatrice
Mme. Noëmie SIMONI, Professeur, Télécom ParisTech	Directrice de thèse

T H È S E

TELECOM ParisTech
école de l'Institut Télécom - membre de ParisTech

À Najwa, Hikmat, Dima et Ayman,
pour leur amour inconditionnel.

Remerciements

À ma directrice de thèse, Madame Noëmie SIMONI, *pour m'avoir accueilli dans son équipe et pour avoir encadré mes travaux de recherche depuis mon stage d'ingénieur et durant mes années de thèse. Je vous remercie de tout mon cœur et je vous assure ma profonde reconnaissance pour votre appui, vos conseils avisés, vos encouragements inlassables et le temps que vous m'avez consacré.*

À Messieurs Omar CHERKAOUI, Ken CHEN, Antoine BOUTIGNON, Elie NAJM, et Mesdames Francine KRIEF et Sylvie VIGNES *pour m'avoir fait l'honneur de participer à mon jury et pour l'intérêt qu'ils ont porté à mon travail.*

À Mesdames Florence BESNARD, Hayette SOUSSOU, Nazha ESSAKKAKI, et Messieurs Bernard CAHEN, Dominique ROUX *pour leur aide, mais surtout pour avoir rendu mon séjour à l'école facile et agréable.*

À tous mes collègues et tous les membres du projet UBIS, *avec qui, j'ai partagé les avancées de nos travaux, ainsi que les moments de stress et de soulagement.*

À tous mes amis *pour m'avoir entouré. Ils étaient toujours présents pour écarter les doutes et partager les joies. Je vous remercie pour votre précieuse présence.*

À Rosy AOUN *pour m'avoir supporté aussi longtemps, en toutes circonstances, pour m'avoir écouté, et pour m'avoir soutenu au niveau émotionnel, moral et professionnel. Je te remercie pour tout ce qu'on a vécu ensemble et qui nous a tant rapproché. C'est vrai que le travail est indispensable au bonheur de l'homme, mais ta présence à ma côté était la vraie source de bonheur dans ce travail.*

À mes parents Hikmat et Najwa, ma sœur Dima et mon frère Ayman *pour leur support, leur encouragement et leurs prières tout au long de ma thèse. Je suis rien sans vous. Je vous aime beaucoup !*

Paris, le 23 Mai 2012
Rachad NASSAR

Résumé

De nos jours, l'avènement de la dérégulation et l'ouverture à la concurrence stimulent les fournisseurs de services à être de plus en plus compétitifs et à attirer de plus en plus d'abonnés afin de faire face aux fortes pressions du marché. Pour ce faire, les fournisseurs d'aujourd'hui favorisent une approche *user-centric* qui consiste à fournir le plus rapidement possible des services orientés utilisateurs. Cette approche *user-centric* gagne de plus en plus d'ampleur suite à l'émergence du contexte de nouvelle génération de réseaux et de services (NGN/NGS). Dans ce contexte où les convergences de réseaux et de services sont omniprésentes, l'utilisateur devient de plus en plus nomade et il réclame l'accès à n'importe quel service, n'importe où, n'importe quand et par n'importe quel moyen. Son but est de composer dynamiquement une session personnalisée de services, dans laquelle converge un ensemble de services multi-domaines (Telco, Web et IT). Ensuite, il désire maintenir la continuité de cette session de services tout au long de sa mobilité spatiale et temporelle. Dans le cadre de cette thèse, nous proposons un ensemble de solutions, pour cette approche *user-centric*, au niveau architectural et fonctionnel afin de gérer la personnalisation et la convergence de services, rendre homogène l'environnement de l'utilisateur, et maintenir la continuité de services. Notre travail de recherche est structuré en trois parties.

Dans la première partie, nous proposons une nouvelle architecture de services, dénommée NGN/NGS Middleware, qui dépasse les problématiques de flexibilité et d'efficacité des architectures verticales monolithiques et des architectures distribuées à couplages forts. Pour ce faire, notre NGN/NGS Middleware suit un modèle architectural horizontal, distribué et trans-organisationnel. Il structure les services selon une approche évènementielle orientée service, tout en s'appuyant sur un modèle de services favorisant l'autogestion, la mutualisation et l'interconnexion par des liens lâches. Par conséquent, notre NGN/NGS Middleware contient un ensemble de services de base qui d'une part maintient la convergence et la personnalisation de services et d'autre part rend transparent le réseau NGN sous-jacent en tenant compte des défis d'hétérogénéité et de mobilité qui le régissent.

Dans la deuxième partie, nous proposons deux solutions pour la gestion de mobilité. La première solution est celle des communautés virtuelles. Elle consiste à anticiper

la coupure de la session de services suite à une dégradation de la QoS (*Quality of Service*) ou d'un mauvais fonctionnement d'un de ses services. Pour ce faire, nous proposons de regrouper des services ubiquitaires ayant la même fonctionnalité et une QoS équivalente dans des communautés virtuelles. Ainsi, lors d'une dégradation d'un service, nous le remplaçons par un autre appartenant à sa communauté, de façon à gérer la continuité tout en tenant compte des préférences fonctionnelles et non fonctionnelles (QoS demandée) de l'utilisateur. La deuxième solution de gestion de mobilité est celle du handover sémantique. Elle permet d'améliorer la QoE (*Quality of Experience*) de l'utilisateur et de réduire le temps de latence, en adaptant la session de services de l'utilisateur à tout changement dans son contexte ambiant. Cette adaptation se fait d'une manière continue et transparente tout au long de la mobilité spatiale et temporelle de l'utilisateur.

Dans la dernière partie, nous pensons apporter une réponse au monde du *cloud computing* en intégrant nos solutions pour gérer les utilisateurs du *cloud*. Ensuite, nous validons la faisabilité de nos travaux à l'aide d'un scénario applicatif qui est mis en œuvre à l'aide d'un démonstrateur à base d'un serveur *GlassFish* où sont intégrés des conteneurs EJB (*Enterprise Java Bean*).

Mots Clés

User-centric, NGN, NGS, architectures de services, telco, web, SOA, EDA, *middleware*, convergence de services, personnalisation de services, gestion de QoS, gestion de mobilité, communautés virtuelles, handover sémantique, hétérogénéité, *cloud computing*.

Abstract

Nowadays, with the advent of deregulation, service providers aim to be more competitive and to attract more subscribers in order to cope with the high market pressure. For this purpose, today's providers support a user-centric approach that consists on quickly providing user oriented services. This user-centric approach becomes more and more significant with the emergence of the next generation networks and services (NGN/NGS) context. Within this context, where network convergence and service convergence are omnipresent, the end-user becomes more nomadic and claims the access to any service, anywhere, anytime and by any means. His goal is to dynamically compose a personalized service session while converging a set of multi-domain services (Telco, Web and IT). Then, he wants to maintain the continuity of this service session throughout his spatial and temporal mobility. Within the scope of this thesis, we propose a set of solutions, for this user-centric approach, on the architectural and functional levels in order to manage the service personalization and convergence, homogenize the end-user's environment, and maintain the service continuity. Our research work is structured into three parts.

In the first part, we propose a novel service architecture, namely the NGN/NGS Middleware, that overcomes the flexibility and efficiency problems occurring in vertical monolithic architectures and in distributed tightly coupled service architectures. For this purpose, our NGN/NGS Middleware adopts an horizontal, distributed and trans-organizational model. The latter structures services according to an event-based service oriented approach, while using a service model that support auto-management, mutualization and loosely coupled interconnection. In consequence, our NGN/NGS Middleware hosts a set of basic services that allow on the first hand, to maintain service personalization and convergence, and on the second hand, to make transparent the underlying NGN network by taking into account its heterogeneity and mobility challenges.

In the second part, we propose two solutions for mobility management. The first solution is based on a novel concept, namely the virtual communities concept. It consists on anticipating the discontinuity of the service session due to a QoS (Quality of Service) degradation or a service malfunction. For this purpose, we propose to combine ubiquitous services, having the same functionality and an equivalent QoS, into vir-

tual communities. Thus, if any service does not fulfill the required QoS, a ubiquitous counterpart that belongs to the same virtual community will dynamically replace it. Therefore, we seamlessly guarantee service continuity while taking into consideration the end-user's functional and non-functional (required QoS) preferences. The second mobility management solution is based on another novel concept, namely the semantic handover concept. It allows to ameliorate the end-user's QoE (Quality of Experience) and to reduce the latency, by adapting the end-user's service session to any change in his ambient context. This adaptation is seamless and continuous throughout the end-user's spatial and temporal mobility.

In the last part, we think we could answer some cloud computing challenges by integrating our solutions to manage cloud users. Then, we validate the feasibility of our work using an applicative scenario that is implemented through a demonstrator based on a GlassFish server and its integrated EJB (Enterprise Java JavaBean) containers.

Keywords

User-centric, NGN, NGS, service architectures, telco, web, SOA, EDA, middleware, service convergence, service personalization, QoS management, mobility management, virtual communities, semantic handover, heterogeneity, cloud computing.

Table des matières

Table des figures

Acronymes

3GPP	*Third Generation Partnership Program*
AJAX	*Asynchronous JavaScript and XML*
ALTO	*Application Layer Traffic Optimization*
API	*Application Programming Interface*
BCP	*Basic Call Process*
BPEL	*Business Process Execution Language*
BS	*Base Station*
BSC	*Basic Station Controller*
CAPEX	*Capital Expenditures*
CN	*Correspondent Node*
CoA	*Care of Address*
CoI	*Community of Interest*
CORBA	*Common Object Request Broker Architecture*
CRUD	*Create, Read, Update and Delete*
CSP	*Cloud Service Provider*
DBS	*Discovery Basic Service*
EDA	*Event-Driven Architecture*
EDGE	*Enhanced Data Rates for GSM Evolution*
EJB	*Enterprise Java Bean*
ESB	*Enterprise Service Bus*
ETSI	*European Telecommunication Standards Institute*
FA	*Foreign Agent*
FBS	*Find Basic Service*

FBSS	*Fast Base Station Switching*
GAN	*Generic Access Network*
Geo-LBS	*Geographic-Location Basic Service*
GPRS	*General Packet Radio Service*
GSL	*Global Service Logic*
GSM	*Global System for Mobile communications*
GSM-CN	*GSM-Core Network*
HA	*Home Agent*
HLSIB	*High-Level SIB*
HoA	*Home Address*
HSS	*Home Subscriber Server*
HTTP	*Hypertext Transfer Protocol*
IETF	*Internet Engineering Task Force*
IMS	*IP Multimedia Subsystem*
INAP	*Intelligent Networks Application Part*
INCM	*Intelligent Network Conceptual Model*
Inter-SP SHS	*Inter Service Providers Semantic Handover Service*
Intra-SP SHS	*Intra Service Provider Semantic Handover Service*
IP	*Internet Protocol*
ISC	*IP Multimedia Service Control*
ISO	*International Organization for Standardization*
IT	*Information Technology*
ITU	*International Telecommunications Union*
JDBC	*Java Database Connectivity*
JMS	*Java Message Service*
JNDI	*Java Naming and Directory Interface*
JPA	*Java Persistence API*
JXTA	*Juxtapose*
LBS	*Location Basic Service*
LTE	*Long Term Evolution*
MCC	*Mobile Cloud Computing*
MDHO	*Micro Diversity Handover*
MIH	*Media Independent Handover*
MIP	*Mobile IP*
MME	*Mobility Management Entity*
MN	*Mobile Node*
MS	*Mobile Station*
MSC	*Mobile Switching Center*

MVNOs	*Mobile Virtual Network Operators*
NGN	*Next Generation Networks*
NGS	*Next Generation Services*
NLR	Nœud, Lien, Réseau
OASIS	*Organization for the Advancement of Structured Information Standards*
OMA	*Open Mobile Alliance*
OPEX	*Operational Expenditures*
OSA	*Open Service Architecture*
OSE	*OMA Service Environment*
P2P	*Peer-to-Peer*
PBS	*Presence Basic Service*
POI	*Point of Initialization*
POR	*Points of Return*
QoE	*Quality of Experience*
QoS	*Quality of Service*
REST	*REpresentational State Transfer*
RI	Réseau Intelligent
RNC	*Radio Network Controller*
RoI	*Return on Investment*
S-CSCF	*Serving Call Session Control Function*
SC	*Service Capabilities*
SE	*Service Element*
SCF	*Service Capability Function*
SCIM	*Service Capability Interaction Manager*
SCS	*Service Capability Server*
SEIB	*Service Elements Interconnection Bus*
SHD	*Semantic Handover Detector*
SIB	*Service Independent Building Blocks*
SIP	*Session Initiation Protocol*
SLA	*Service Level Agreement*
SLMA	*Service Level Management Authority*
SOA	*Service-Oriented Architecture*
SOAP	*Simple Object Access Protocol*
TTM	*Time to Market*
UBIS	*"User-centric"* : uBiquité et Intégration de Services
UBS	*Ubiquity Basic Service*

UDDI	*Universal Description, Discovery, and Integration*
UE	*User Equipment*
UMA	*Unlicensed Mobile Access*
UmaNC	*UMA Network Controller*
UMTS	*Universal Mobile Telecommunications System*
URI	*Unified Resource Identifier*
URL	*Unified Resource Locator*
VCCU	*Ubiquity-based Virtual Connectivity Community*
VECU	*Ubiquity-based Virtual Equipment Community*
VPCN	*Virtual Private Connectivity Network*
VPEN	*Virtual Private Equipment Network*
VPSN	*Virtual Private Service Network*
VSCU	*Ubiquity-based Virtual Service Community*
VSCU-L&P	*Ubiquity, Location and Provider based Virtual Service Community*
VSC-U&L	*Ubiquity and Location based Virtual Service Community*
VSC-U&P	*Ubiquity and Provider based Virtual Service Community*
W3C	*World Wide Web Consortium*
W-CDMA	*Wideband Code Division Multiple Access*
WiMAX	*Worldwide Interoperability for Microwave Access*
WS-CDL	*Web Services Choreography Description Language*
WSDL	*Web Services Description Language*
XaaS	*Everything as a Service*
XML	*eXtensible Markup Language*

1 Introduction Générale

1.1 Contexte d'étude

De nos jours, la vue économique et stratégique du marché se manifeste essentiellement par une interaction entre les utilisateurs et les fournisseurs de services selon le principe de croisement des intérêts. D'une part, les utilisateurs cherchent à avoir la meilleure QoE (*Quality of Experience*) pour leurs sessions. Pour cela, ils s'abonnent au près des fournisseurs qui répondent le mieux à leurs demandes, leurs besoins et leurs préférences. D'autre part, les fournisseurs de services cherchent à être compétitifs en face des fortes pressions du marché. Ils s'investissent à proposer, de plus en plus rapidement, des services de plus en plus complexes et de plus en plus orientés utilisateurs. Cela leur permet de réduire considérablement le TTM (*Time to Market*) tout en maximisant leurs nombres d'abonnés et par conséquent leurs RoI (*Return on Investment*).

Suite à l'évolution technologique qu'a subie le monde des télécommunications, de nouveaux facteurs entrent en jeu. Ces derniers deviennent déterminants du succès des fournisseurs de services et primordiaux vis-à-vis de la QoE des utilisateurs. Dans le cadre de cette thèse, les facteurs techniques qui nous intéressent se manifestent dans un contexte technologique où convergent les problématiques des besoins *user-centric* (voir Sous-Section 1.1.1), l'architecture intégrée NGN/NGS (voir Sous-Section 1.1.2) et la mobilité (voir Sous-Section 1.1.3).

1.1.1 User-centric

Au cours des années, le contexte technologique a évolué vers une approche *user-centric* tout en passant, comme le montre la Figure 1.1, par différentes approches : la première, appelée *system-centric*, implique que le *hardware* soit au centre de l'architecture ; la deuxième, appelée *application-centric*, considère l'application comme le point focal qui conditionne les demandes de l'utilisateur ; et la troisième, appelée *network-centric*, se concentre sur les réseaux et sur les supports de connexions. Contrairement aux autres approches, le contexte *user-centric* met l'utilisateur au centre de l'architecture, et impose ainsi aux fournisseurs et aux opérateurs d'adapter leurs services, leurs produits *hardware* et leurs réseaux de façon à être toujours "au service" de cet utilisateur. Ce

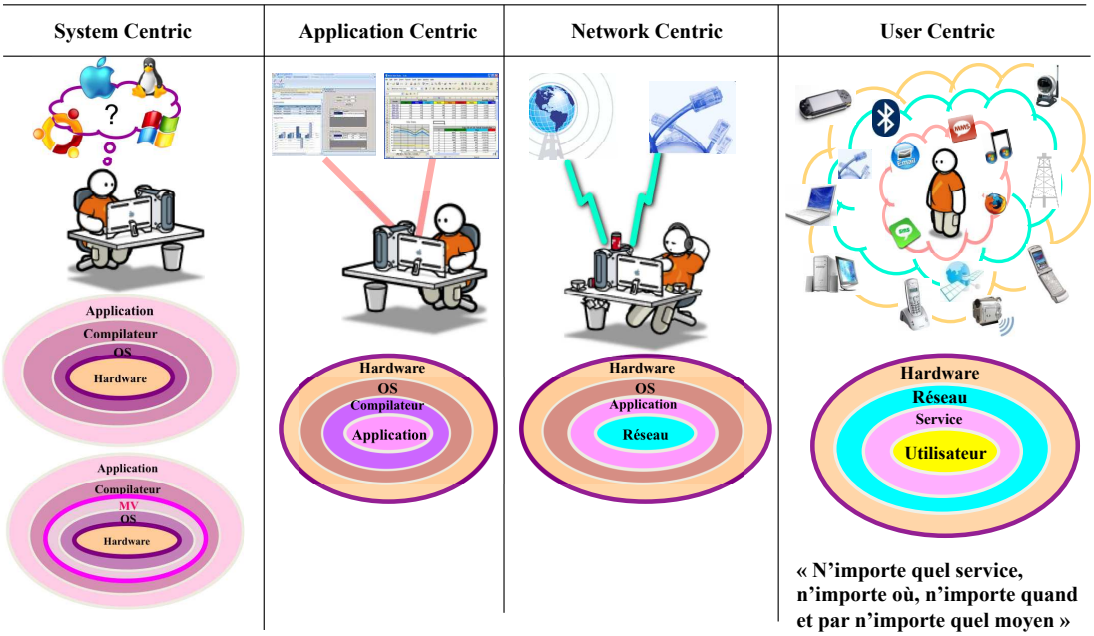

Fig. 1.1: Evolution du contexte technologique vers l'approche *user-centric*.

dernier n'est donc plus obligé de se plier aux différentes contraintes de traitement et de connexion. Il réclame l'accès à n'importe quel service, n'importe où, n'importe quand et par n'importe quel moyen sans aucune restriction technique et/ou économique.

Dans le cadre de cette approche *user-centric*, tout utilisateur désire établir dynamiquement une session unique et personnalisée. La session est vue comme une mise en relation temporelle entre l'utilisateur et l'ensemble du système (depuis son terminal jusqu'à la plate-forme de services, tout en passant par les composants des réseaux d'accès et du réseau cœur). Elle est personnalisée, ce qui permet à tout utilisateur de composer sa propre session de services, à l'aide de différents types de services (Telco, Web et IT (*Information Technology*)) et selon une logique adéquate à son contexte ambiant, ses préférences fonctionnelles, ainsi que ses préférences non-fonctionnelles, c.à.d. la QoS demandée. Elle est unique, ce qui nécessite sa maintenance en temps réel et d'une manière dynamique et transparente vis-à-vis de l'utilisateur. Ceci permet ainsi d'avoir une session par utilisateur tout au long de son déplacement spatial et temporel, et malgré les modifications qui sont dûes à la personnalisation de services.

1.1.2 Architecture NGN/NGS

De nos jours, le NGN (*Next Generation Networks*) [41][43][51] est considéré comme la plus grande évolution dans le monde des télécoms, et comme étant un premier pas vers une approche *user-centric* dans le monde économique et technologique d'aujourd'hui. Ce nouveau concept modélise d'une manière générique les nouvelles architectures de réseaux de communication, qui reposent sur l'indépendance et la séparation formelle entre les niveaux accès, transport, contrôle et service. Selon la Figure 1.2, l'architecture NGN est constituée d'un ensemble de couches horizontales gérées indépendamment :

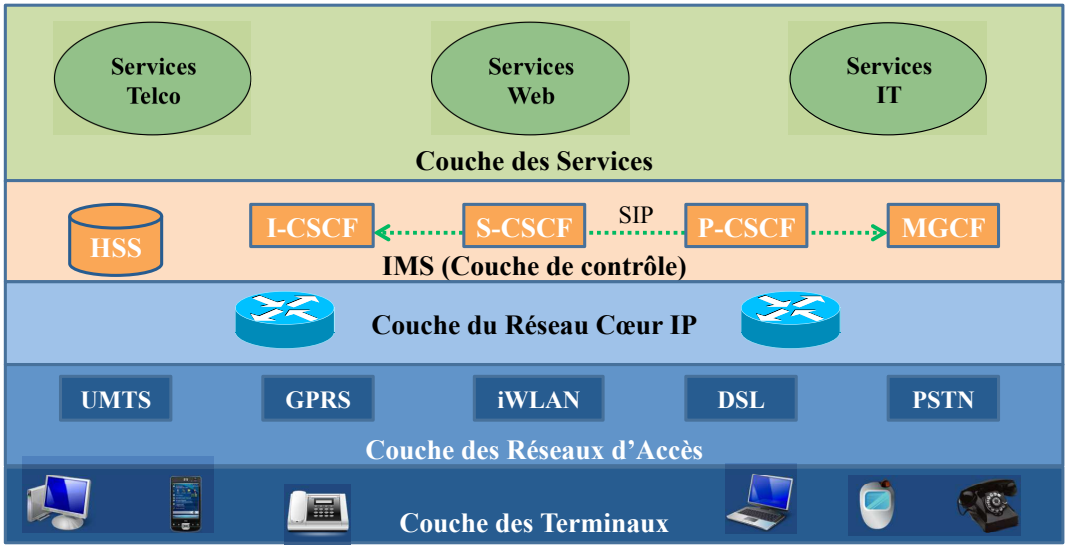

Fig. 1.2: Architecture NGN en couches.

- La couche des terminaux : elle contient un ensemble de terminaux de natures et de capacités différentes. Nous pouvons ainsi trouver dans l'environnement de l'utilisateur des ordinateurs fixes et portables, des téléphones fixes, des téléphones mobiles supportant différentes technologies cellulaires, des téléphones IP, etc.
- La couche d'accès : elle contient un ensemble de technologies d'accès de natures et de caractéristiques différentes. L'utilisateur peut ainsi accéder à ses services en utilisant des réseaux d'accès de différentes technologies comme les réseaux cellulaires, les réseaux WiFi, les réseaux téléphoniques commutés, et les différentes technologies des réseaux filaires.
- La couche de transport : elle est dominée par le protocole IP (*Internet Protocol*) depuis l'avènement de l'*Internet*. En effet, l'approche *best-effort* adoptée par IP permet un transfert rapide des données, ce qui a favorisé l'usage d'IP comme protocole de transport. Cette couche de transport vient s'interfacer au-dessus de la couche d'accès, permettant ainsi la convergence du trafic provenant de différents réseaux d'accès dans une seule couche de transport IP.
- La couche de contrôle : elle est basée sur la technologie IMS (*IP Multimedia Subsystem*) [43] qui introduit un environnement de contrôle de session. IMS permet ainsi d'établir, de modifier et de libérer des sessions multimédias en se basant sur le protocole de signalisation SIP (*Session Initiation Protocol*). De plus, IMS introduit plusieurs fonctions et interfaces pour invoquer les services, contrôler le trafic IP transporté, contrôler les politiques d'accès aux services, maintenir les profils des utilisateurs et de leurs services, garantir la sécurité d'accès aux services, garantir un environnement OAM&P (Opération, Administration, Maintenance et Provisionnement), etc.
- La couche des services : elle contient un ensemble de services de natures différentes (Telco, Web ou IT).

Grâce à cette séparation en différentes couches, NGN permet aux fournisseurs de

services, aux opérateurs et aux fournisseurs d'équipements de créer, gérer et faire évoluer leurs produits indépendamment les uns des autres. Cependant, avoir une vision globale du marché reste toujours un facteur important dans la stratégie économique des fournisseurs et des opérateurs. En outre, autre que la séparation en couches horizontales, la convergence de réseaux constitue le deuxième atout primordial de l'évolution NGN. Grâce à ce nouveau concept, l'utilisateur peut utiliser n'importe quel terminal et n'importe quel réseau d'accès afin d'accéder à ses services. Cependant, en face à cette liberté d'accès garantie par ce cadre NGN multi-accès et multi-terminaux, des défis NGN surgissent, comme la continuité de la session, la QoS de bout-en-bout et l'hétérogénéité de l'environnement complexe entourant l'utilisateur. La gestion de ces défis est ainsi primordiale pour avoir une meilleur QoE.

En parallèle à cette évolution des architectures de réseau de communication vers un nouveau contexte NGN, une nouvelle ère de services voit le jour. En effet, aujourd'hui, des fournisseurs de services sans réseaux (ex. *Google, Amazon, Skype,* etc.) rendent leurs services sans s'appuyer sur aucune architecture réseau fournie par un opérateur télécom, mais plutôt en se basant sur des architectures techniques issues du monde Web et du monde IT. Ces acteurs cherchent à augmenter leurs parts du marché en adoptant un nouveau modèle économique. Ce dernier consiste à surmonter la concurrence du marché en attirant le plus d'abonnés. Pour ce faire, ils ne se contentent plus de traiter les abonnés comme des simples clients d'une offre d'accès. Au contraire, ils les considèrent comme des utilisateurs ayant des besoins et des préférences qui doivent être pris en compte dans leurs écosystèmes. Avec ce nouveau contexte de services orientés utilisateur, une nouvelle architecture semble nécessaire pour structurer cette nouvelle vision de services de nouvelle génération, notés NGS (*Next Generation Services*). Pour répondre aux besoins de l'approche *user-centric*, l'architecture NGS voulue doit dépasser deux défis primordiaux :

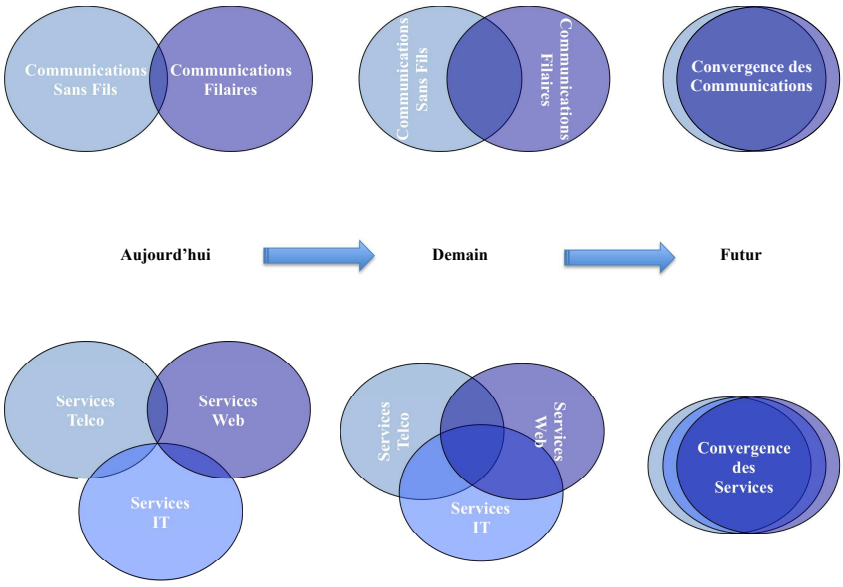

Fig. 1.3: Evolution de la convergence au niveau réseau et service.

- **La convergence de services :** par analogie à la convergence de réseaux qui permet à l'utilisateur d'accéder à un même service en utilisant n'importe quel types de réseaux d'accès, la convergence de services consiste à permettre à tout utilisateur d'avoir dans une même session de services des caractéristiques multi-domaines et multi-fournisseurs, c.à.d., des services trans-organisationnels, appartenant à différents domaines (Telco, Web et IT). La Figure 1.3 montre cette analogie entre la convergence au niveau réseau et celle au niveau service.
- **La personnalisation de services :** dans le cadre de convergence de services, l'utilisateur doit avoir la liberté de composer sa propre session de services sans aucune barrière technologique ou économique. Cette composition de services s'appuie ainsi sur des services multi-domaines, multi-fournisseurs et multi-caractéristiques, qui coopèrent entre eux, afin de répondre aux préférences et aux besoins de l'utilisateur. Il faut noter que cette personnalisation dans la composition de services intervient non seulement lors de la phase de création de la session de services de l'utilisateur, mais aussi lors de la phase de gestion. En effet, la session de services doit être maintenue en temps réel et d'une manière dynamique et transparente vis-à-vis de l'utilisateur, de façon à toujours contenir les services qui respectent le plus les préférences fonctionnelles de l'utilisateur, la QoS demandée ainsi que son contexte ambiant.

Au cours des années, plusieurs tentatives ont été faites pour structurer la couche service, cependant, aucune d'entre elles n'a pu surmonter les défis évoqués précédemment. D'après la Figure 1.4, nous constatons que la conception d'une nouvelle architecture de services a évolué d'une architecture en silos orientée traitement, vers une architecture distribuée orientée objet, pour devenir aujourd'hui une approche orientée service.

L'approche monolithique des architectures en silos était pendant longtemps considérée la solution la plus efficace, la plus performante et la plus optimale pour l'intégration des services. Cette approche verticale est basée sur un ensemble de silos indépendants dont chacun est responsable d'un ensemble de ressources, d'information et d'applications distinctes. Ces systèmes dédiés pour l'intégration de services ont été fortement appliqués dans l'industrie, cependant, ils ne peuvent pas répondre aux nouveaux besoins des architectures d'aujourd'hui. Du point de vue des fournisseurs de services, cette approche monolithique verticale pose des problèmes au niveau OPEX (*Operational Expenditures*) comme au niveau CAPEX (*Capital Expenditures*). En effet, cette

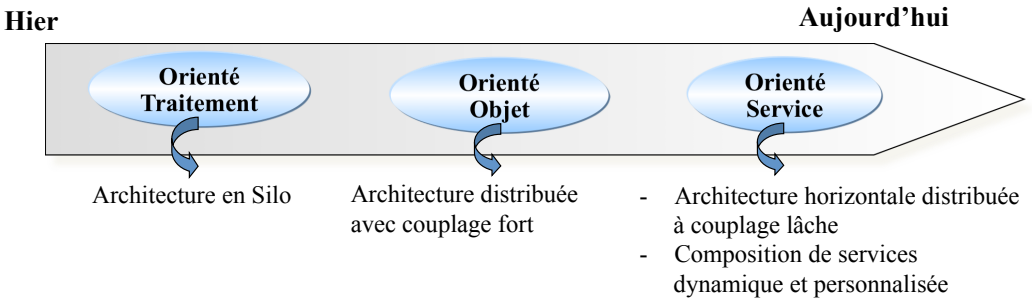

Fig. 1.4: Evolution de la conception des architectures de services.

approche verticale est basée sur un ensemble d'éléments interconnectés selon un couplage fort, ce qui impose un remplacement du système tout entier pour toute tentative de maintenance et de mise à jour de l'architecture. Cela nécessite plus d'opérations et un plus grand coût traduit par un OPEX et un CAPEX plus élevés. En outre, cette architecture verticale consiste à isoler les fonctions, les données, les ressources et les politiques correspondant à un silo des autres systèmes qui pourraient s'en servir. Par conséquent, l'effort effectué pour la conception, la modulation et la création des différents éléments d'un silo ne peut pas être réutilisé par d'autres systèmes, ce qui implique la création, l'intégration et la maintenance indépendante de nouveaux éléments architecturaux. Cela augmente aussi l'OPEX et le CAPEX, et diminue le rendement de certaines ressources en empêchant le partage de leurs capacités même si ces dernières restent inutilisées pendant une certaine période. D'autre part, cette approche monolithique cause aussi des problèmes au niveau de l'interaction avec l'utilisateur dont la QoE se dégrade suite au déploiement indépendant des applications qui mène vers une réutilisation inconsistante de ses informations et de ses préférences. De plus, les architectures en silos ne prennent pas en considération les défis qui dérivent des réseaux NGN sous-jacents, comme la continuité de la session de l'utilisateur et l'hétérogénéité de son environnement ambiant.

Afin de dépasser les problèmes causés par l'approche orientée traitement qui produisait des applications séquentielles et monolithiques, une nouvelle architecture a été conçue. Cette dernière est une architecture distribuée orientée objet. Elle se présente comme étant une solution mieux adaptée aux exigences des fournisseurs de services. Elle introduit la notion de coopérations multi-parties qui permettent de pallier les problèmes de flexibilité des architectures en silos. Elle réduit le CAPEX et l'OPEX en rendant les systèmes d'intégration plus agiles et plus adaptatifs vis-à-vis des changements au niveau des besoins métiers. Ces nouveaux systèmes distribués orientés objet ont été à l'origine de la nouvelle génération des systèmes de communications émergents, comme le commerce électronique, l'imagerie médicale distribuée, etc. Cependant, ce traitement distribué est fortement influencé par des contraintes liées à la structuration des objets qui induit des liens forts. En effet, malgré l'intelligence et la sémantique introduite par cette architecture, le couplage fort entre les objets cause une certaine dépendance qui rend les fournisseurs incapables de s'adapter aux nouveaux besoins des utilisateurs, comme la mobilité, les préférences fonctionnelles, la QoS et la prise en compte du contexte ambiant.

Pour cette raison, nous nous orientons vers la notion de service qui préconise des liens lâches. Le périmètre fonctionnel de cette nouvelle vision de services se définit par rapport à des spécifications d'utilisation indépendamment des enchaînements. Le but est de permettre à l'utilisateur de composer sa propre session de services à l'aide de services composables à couplages lâches, et de permettre aussi aux fournisseurs de services de se servir de leurs systèmes distribués d'une manière plus efficace. Par conséquent, le contexte NGS dans lequel se situe notre travail est celui d'une architecture distribuée, horizontale, orientée services à couplage lâches, garantissant la convergence et la personnalisation des services.

1.1.3 Mobilité

L'évolution rapide des réseaux NGN dans l'industrie de la communication assure une grande couverture et une diversité de technologies sur les offres d'accès accompagnées d'une hétérogénéité au niveau des terminaux utilisés, ce qui permet à l'utilisateur de se déplacer au gré de ses besoins sans aucune limitation. Ce contexte NGN favorise ainsi l'approche *user-centric* et rend l'utilisateur d'aujourd'hui de plus en plus nomade et de plus en plus exigeant au niveau de sa QoE. Par conséquent, dans le cadre de l'approche "*Every Service for Everyone, Anywhere, Anytime and Anyhow*", tout utilisateur désire conserver sa session de bout-en-bout qui lui permet d'accéder d'une manière continue à ses services, quelque soit le terminal, le réseau d'accès, le réseau cœur, et le fournisseur de services choisis. De plus, il réclame une session continue qui prend bien en considération ses préférences fonctionnelles et sa QoS de bout-en-bout demandée. Dans ce nouveau contexte NGN et *user-centric*, les exigences au niveau mobilité deviennent des graves défis qui augmentent la complexité de la gestion des ressources. Afin de dépasser ce problème de continuité de session, nous devons préciser les différents évènements qui peuvent surgir durant la session de l'utilisateur et peuvent impacter sa continuité. Nous les classifions sous quatre types de mobilités :

- **Mobilité du terminal :** elle représente le passage du terminal d'un point d'accès réseau à un autre. En effet, grâce à l'évolution technologique au niveau des réseaux d'accès filaires et des réseaux d'accès sans fils, et grâce à l'apparition des terminaux multi-technologies, l'utilisateur d'aujourd'hui est devenu capable d'accéder à ses services lors de son déplacement géographique, tout en se servant d'un même terminal qui commute d'un réseau d'accès à un autre. Pour cette raison, ce type de mobilité spatiale pose un défi vis-à-vis de la continuité de la session.
- **Mobilité de l'utilisateur :** elle représente le passage d'un utilisateur d'un terminal à un autre. En effet, de nos jours, tout utilisateur se trouve dans un environnement hétérogène qui contient des terminaux très diversifiés. Par conséquent, l'utilisateur est devenu capable d'accéder à ses services tout en passant d'un terminal à un autre. Pour cela, ce type de mobilité spatiale pose un défi vis-à-vis de la continuité de la session.
- **Mobilité du réseau :** elle représente le déplacement d'un routeur et par conséquent du plan de routage du réseau associé. Les machines virtuelles devraient

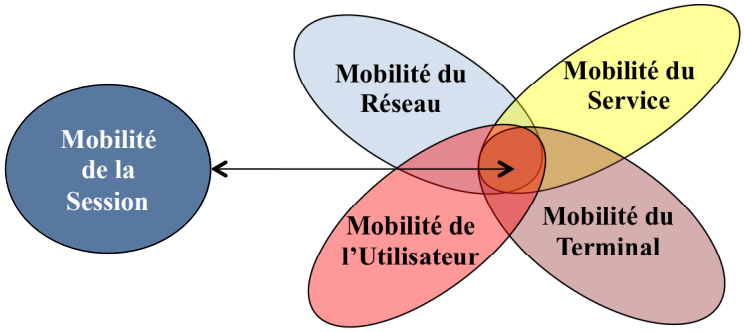

Fig. 1.5: Différents types de mobilité.

simplifier, grâce à leurs configurations dynamiques, la prise en compte de cette mobilité du réseau.

- **Mobilité du service :** elle est dûe essentiellement au principe d'ubiquité de services qui consiste à avoir des services ubiquitaires, c.à.d. des services ayant la même fonctionnalité et une QoS équivalente. Par conséquent, le principe de mobilité de services représente le remplacement d'un service demandé par l'utilisateur par un autre service qui lui soit ubiquitaire. D'après l'approche trans-organisationnelle que nous supportons, le service remplaçant peut appartenir à un autre fournisseur de services. En effet, ce type de mobilité devient un défi de plus en plus important à gérer, surtout avec le *booming* du marché des services et l'augmentation de la concurrence qui incite les fournisseurs à déployer des services ubiquitaires pour mieux répondre aux besoins de leurs utilisateurs.

Afin de maintenir la continuité de la session tout en respectant la QoS de bout-en-bout et les préférences fonctionnelles demandées par l'utilisateur, il est primordiale de gérer ces quatre types de mobilité. Comme le montre la Figure 1.5, la mobilité de la session résulte de la gestion de ces quatre types de mobilité. Cependant, dans le cadre de cette thèse, nous nous intéressons seulement à la mobilité spatiale au niveau terminal et utilisateur, ainsi que la mobilité de services, et nous négligeons la mobilité du réseau.

D'autre part, selon l'approche *user-centric*, la satisfaction de l'utilisateur n'est pas seulement basée sur la continuité de sa session et sur la prise en compte de ses préférences. La prise de conscience du contexte ambiant de l'utilisateur [13] est aussi un facteur essentiel dans l'amélioration de la QoE de cet utilisateur. Elle consiste à adapter la session de l'utilisateur à son nouveau contexte ambiant. Ce dernier consiste en un ensemble d'informations temps réel qui caractérisent l'environnement de l'utilisateur. Ainsi, pour une meilleure QoE et pour une gestion de mobilité plus efficace, les informations ambiantes doivent être prises en considération. Dans le cadre de cette thèse, nous considérons que les changements dans le contexte ambiant d'un utilisateur sont dûs essentiellement à une mobilité spatiale (mobilité du terminal ou mobilité de l'utilisateur). En effet, durant la mobilité temporelle de l'utilisateur, c.à.d. durant sa session continue, un nouveau contexte ambiant est découvert dynamiquement suite à toute commutation d'un terminal à un autre, ou d'un point d'accès réseau à un autre.

Afin d'améliorer la prise en conscience du nouveau contexte ambiant de l'utilisateur et afin de traiter de façon plus efficace les nouvelles informations ambiantes, le concept des réseaux ambiants [29][45] a été conçu dans les années 2002/2003. Ce concept se résume comme une architecture ouverte d'interconnexion de réseaux hétérogènes qui se composent et se décomposent dynamiquement afin de permettre la mise en relation automatique et temps réel des terminaux relatifs aux utilisateurs. Par conséquent, lors du déplacement de l'utilisateur, par exemple lorsqu'il entre dans une gare, son réseau ambiant constitué des équipements qui sont à sa portée s'adapte en se composant au réseau ambiant déjà existant dans cette gare. Il faut noter que la couverture des réseaux ambiants peut être petite, comme elle peut être aussi grande (ex. de l'ordre d'une ville). En effet, le concept des réseaux ambiants assure plusieurs avantages : l'interconnexion des réseaux d'accès et des équipements hétérogènes, la convergence fixe-mobile,

la gestion de la transparence de la mobilité, la gestion du contexte de l'utilisateur et la sécurité. Cependant, cette solution reste toujours limitée aux réseaux et aux terminaux, et elle néglige toute prise en conscience du contexte de l'utilisateur au niveau de la couche service. Afin de dépasser ce problème, nous considérons l'extension du concept de réseaux ambiants pour prendre en compte la couche service. Ainsi, pour chaque nouvelle situation ambiante de l'utilisateur, dûe à une mobilité spatiale, de nouveaux services sont découverts. Certains de ces services viennent s'additionner à la session de services courante de l'utilisateur, et d'autres s'annoncent comme des candidats ubiquitaires à certains services dans la session de l'utilisateur, ce qui permet le remplacement de ces services non-ambiants par d'autres appartenant à la nouvelle zone ambiante de l'utilisateur.

Pour atteindre ce but, les fournisseurs de services doivent offrir des distributions de services à grandes échelles. Ainsi, chaque fournisseur de services, en prenant en considération les statistiques du marché de services et sa stratégie économique adaptée pour lutter contre la compétitivité au sein de ce marché, doit déployer ses services ubiquitaires distribués sur différentes zones géographiques, de façon à couvrir la majorité du territoire de ses utilisateurs ciblés. Si pour des raisons budgétaires ou autres, un fournisseur de services n'est pas capable de couvrir toute la zone ciblée, il peut toujours négocier et signer des contrats de *peering* avec d'autres fournisseurs offrant des services ubiquitaires.

Cette distribution à grande échelle des services ubiquitaires, ainsi que ces contrats de fédération entre les fournisseurs de services, assurent plusieurs avantages. D'une part, elles permettent aux fournisseurs de services de respecter toujours les contrats de SLA (*Service Level Agreement*) signés avec leurs utilisateurs, ce qui mène à capter de plus en plus d'abonnées et par suite une augmentation du RoI de ces fournisseurs. D'autre part, au niveau utilisateur, ces approches respectent le contexte *user-centric*. En effet, elles tiennent compte du contexte ambiant de l'utilisateur en ajoutant de nouveaux services ambiants dans la session de services, et en remplaçant les services non-ambiants provisionnés par des nouveaux services ambiants qui lui sont ubiquitaires. En outre, ces approches permettent aussi de connecter, d'une manière transparente, tout utilisateur aux plates-formes de services les plus proches, tout au long de sa mobilité spatiale. Cela optimise ainsi la QoS de bout-en-bout en diminuant la latence. Ce facteur dépend essentiellement sur la distance séparant un utilisateur de son service et il a une grande influence sur la QoE de l'utilisateur. Par exemple, une latence réseau de quelques millisecondes devient critique dans le cas des services à interaction temps réel, comme par exemple les applications immersives [15], et peut ainsi causer une dégradation importante dans l'expérience de l'utilisateur.

1.2 Périmètre de la thèse

Après avoir décrit les différents contextes qui guident les avancements technologiques d'aujourd'hui et ceux de l'avenir, nous délimitons, comme le montre la Figure 1.6, notre périmètre d'étude dans cette thèse à l'intersection des trois contextes déjà évoqués : l'approche *user-centric*, l'architecture NGN/NGS et la mobilité.

En effet, nous avons besoin de repenser l'architecture globale de façon à supporter la sémantique spécifique de nos attentes. Cette structure d'ensemble doit faire ainsi converger les contextes NGN et NGS tout en tenant compte de l'approche *user-centric* et de la mobilité de la session. Notamment, l'architecture de services voulue doit répondre aux besoins suivants :

- La personnalisation des services pour une composition dynamique des services à couplages lâches selon les besoins de l'utilisateur.
- L'adaptation temps réel de cette composition de services à tout type d'évènement, tout en restant dynamique et transparente vis-à-vis de l'utilisateur.
- La convergence de services pour une intégration horizontale entre des services appartenant à différents domaines (Telco, Web et IT).
- La convergence de la couche service avec la couche NGN sous-jacente pour une intégration verticale qui tient compte des défis d'hétérogénéité et de mobilité qui régissent la convergence réseau.
- La maintenance d'une session unique et continue, qui tient compte des préférences fonctionnelles et non-fonctionnelles (QoS demandée) de l'utilisateur, ainsi que son contexte ambiant.

Il faut noter que dans le cadre de cette thèse, nous considérons seulement les aspects architectural et fonctionnel lors de la conception de notre "écosystème" qui répond aux besoins des fournisseurs de services et des utilisateurs. Les autres aspects sont étudiés par d'autres chercheurs au sein de notre groupe de recherche.

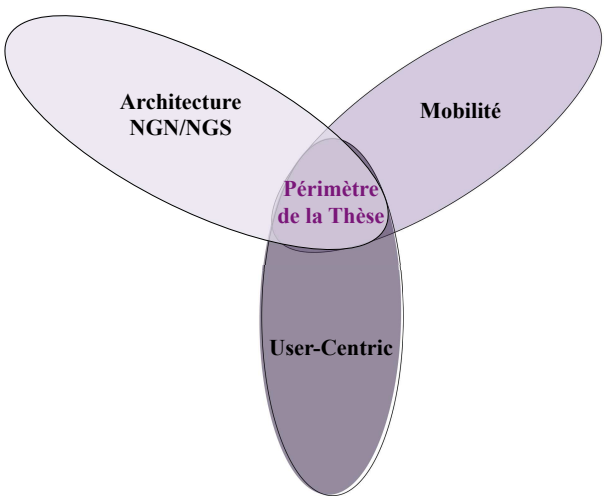

Fig. 1.6: Périmètre de la thèse.

1.3 Problématiques

Avec l'avènement de la dérégulation et l'ouverture à la concurrence, l'approche *user-centric* commence à prendre de l'ampleur. Ainsi, les fournisseurs et les opérateurs cherchent à augmenter leurs RoI et leurs parts du marché en introduisant le plus rapidement possible des services qui attirent le plus grand nombre d'utilisateur et qui répondent le mieux à leurs besoins.

Un **premier verrou** se pose : *Comment concevoir le service pour mieux répondre aux défis des fournisseurs de services et aux attentes de leurs utilisateurs ? Quel modèle de service permettrait-il la personnalisation de la session de services de l'utilisateur, ainsi que la convergence dans une même session des services appartenant aux différents domaines Telco, Web et IT ?*

Par ailleurs, cette évolution des services introduira une évolution de l'architecture qui structure la sémantique de l'écosystème. Cette architecture est voulue horizontale, convergente, orientée service, auto-organisée et autogérée. Elle doit aussi cohabiter avec la couche NGN sous-jacente.

Nous identifions notre **deuxième verrou** : *Comment structurer les services de façon à favoriser l'autonomie, la distribution multi-domaines et la coopération trans-organisationnelle ? Quels sont les processus à introduire afin de supporter la composition personnalisée et convergente des services ? Comment établir dynamiquement, pour chaque utilisateur, une session unique qui tient compte de ses préférences fonctionnelles et non-fonctionnelles (c.à.d. sa QoS demandée) ? Comment rendre homogène l'environnement de l'utilisateur ainsi que la vision de bout-en-bout des fournisseurs et des opérateurs ?*

De même, dans ce nouveau contexte NGN/NGS et *user-centric*, l'utilisateur devient de plus en plus nomade et désire accéder à sa session de services n'importe quand, n'importe où et avec n'importe quel moyen. Par conséquent, la continuité de la session est mise en question. En effet, un des facteurs qui menacent la continuité de la session est celui de la dégradation de la QoS des services provisionnés.

Ainsi, nous identifions un **troisième verrou** à lever : *Comment maintenir, d'une manière transparente, la continuité de la session de services lors d'une dégradation de la QoS ou un mauvais fonctionnement d'un des services pré-provisionnés pour un utilisateur donné ?*

De plus, la satisfaction de l'utilisateur n'est pas limitée à la continuité de sa session de services et au respect du contrat de QoS de bout-en-bout. Pour un meilleur rendement, les fournisseurs de services doivent penser à adapter la session de l'utilisateur aux changements au niveau du contexte ambiant de ce dernier. Par suite, chaque utilisateur sera relié aux ressources les plus proches ce qui minimise la latence et améliore sa QoE. Ce défi est traité au niveau réseau par les réseaux ambiants, mais aucune tentative n'est faite au niveau service.

Le **quatrième verrou** apparaît ainsi : *Comment prendre en considération le contexte*

ambiant de l'utilisateur, tout au long de sa mobilité spatiale et temporelle, tout en assurant une continuité sans couture de sa session de services ?

1.4 Organisation

Nous organisons le rapport de thèse en douze chapitres répartis en trois grandes parties :

- Le Chapitre 1, est cette introduction générale qui a pour objectif de décrire les contextes technologiques dans lesquels se situe notre thèse, ainsi que les motivations et les problématiques. Nous avons commencé par présenter les contextes de l'approche *user-centric*, des architectures convergentes NGN/NGS et de la mobilité. Ensuite, nous avons délimité le périmètre de cette thèse à l'intersection de ces trois contextes technologiques. Puis, nous avons identifié les verrous que nous proposons de lever.

C'est alors que commence la **Partie I** dans laquelle nous proposons notre solution pour l'architecture de services. Cette partie est répartie en quatre chapitres :

- Le Chapitre 2, introduit et donne la structure de la Partie I.
- Le Chapitre 3, donne une synthèse de notre étude bibliographique au niveau des architectures de services existantes. Avant d'envisager les nouvelles solutions concernant l'architecture de services, nous avons analysé les architectures de services qui existent au niveau Telco, Web et IT, et nous avons discuté leurs avantages et leurs inconvénients face aux nouveaux besoins qui régissent le nouveau contexte technologique NGN/NGS et *user-centric*.
- Le Chapitre 4, consigne nos propositions au niveau de l'architecture de services. La Section 4.1 présente les caractéristiques de notre nouveau modèle architectural qui permet de lever le premier verrou identifié. En effet, nous adoptons une approche évènementielle orientée service, qui se base sur un nouveau modèle de service. Ce dernier supporte la nouvelle vision de services NGS et répond aux besoins de convergence et de personnalisation de services. De plus, nous adoptons une approche horizontale sous forme d'un *Middleware* de services afin de favoriser l'autonomie, la distribution multi-domaines et la coopération trans-organisationnelle. Ce modèle divise l'architecture en trois parties : les services de bases, le bus d'interconnexion et les services exposables. Ensuite, nous proposons dans la Section 4.2 la nouvelle architecture de services notée NGN/NGS Middleware qui permet de lever le deuxième verrou identifié. Elle comporte des services exposables qui seront présentés dans les catalogues des fournisseurs. De plus, elle comporte un support d'interconnexion et des services de bases qui supportent la composition personnalisée des services et qui permettent d'établir dynamiquement, pour chaque utilisateur, une session unique qui tient compte de ses préférences fonctionnelles et non-fonctionnelles. En outre, afin de garantir la cohabitation des contextes NGN et NGS pour une meilleur QoE, nous propo-

sons des services de base pour gérer les défis provenant de la convergence réseau, comme l'hétérogénéité et la mobilité.
- Le Chapitre 5, conclut la Partie I et fait le lien avec la Partie II qui suit.

La **Partie II** est consacrée à notre solution pour la gestion de mobilité. Cette partie est répartie en quatre chapitres :

- Le Chapitre 6, introduit et donne la structure de la Partie II.
- Le Chapitre 7, donne une synthèse de notre étude bibliographique au niveau des solutions existantes pour la gestion de la mobilité. Avant d'envisager les nouvelles solutions concernant la gestion de mobilité, nous avons analysé les solutions qui existent au niveau des réseaux d'accès (handovers horizontaux et handovers verticaux) et du réseau cœur IP (*Mobile IP*). Ensuite, nous avons discuté leurs avantages et leurs inconvénients dans le cadre de gestion de la mobilité.
- Le Chapitre 8, consigne nos propositions au niveau de la gestion de la mobilité. La Section 8.1 permet de lever le troisième verrou identifié, en présentant notre solution de gestion de la continuité de la session de services, suite à une dégradation de la QoS ou à un mauvais fonctionnement d'un des services sélectionnés. Elle se base sur le principe de gestion par communautés virtuelles qui à leur tour s'appuient sur le principe d'ubiquité de services. En outre, la Section 8.2 permet de lever le quatrième verrou identifié, en présentant notre deuxième solution de gestion de mobilité qui consiste à étendre le principe de handover qui existe au niveau réseaux d'accès et réseau cœur, et de l'appliquer au niveau service. Nous proposons ainsi le principe de handover sémantique qui permet d'adapter la session de services de l'utilisateur aux changements dans son contexte ambiant, tout au long de sa mobilité spatiale. Cela permet aussi de minimiser la latence et d'améliorer la QoE. Pour chacune de ces deux solutions, nous présentons les aspects architectural et fonctionnel, ainsi qu'une mise en œuvre qui aide à la compréhension des principes évoqués.
- Le Chapitre 9, conclut la Partie II et fait le lien avec la Partie III qui suit.

Dans la **Partie III**, nous présentons la valorisation et la faisabilité de nos propositions. Cette partie est répartie en deux chapitres :

- Le Chapitre 10, consigne la valorisation de nos travaux dans le contexte du *cloud computing*. En effet, notre architecture peut aider à mieux structurer les plates-formes de services dans le cloud de façon à tenir compte des défis aux niveaux des modèles de services proposés, de la gestion de la QoS et de la gestion de la mobilité.
- Le Chapitre 11, montre la faisabilité de nos travaux. La Section 11.1 présente un scénario applicatif complet qui met en œuvre l'ensemble des propositions de gestion évoquées. Ensuite, la Section 11.2 présente le démonstrateur composé du cœur IMS *Open IMS Core*, du serveur d'application *GlassFish*, de la base de connaissances *Oracle* et des routeurs virtuels *VirtuOR*.

Enfin,

– Le Chapitre 12 nous permet de conclure notre travail et de discuter des éventuelles perspectives.

A la fin de ce rapport de thèse, nous présentons la liste des publications internationales ainsi que la liste des délivrables dans le cadre du projet national UBIS ("User-centric" : uBiquité et Intégration de Services).

Première partie

Architecture de services

2 Introduction

Avec l'avènement de la nouvelle génération de réseau, et surtout avec l'évolution des réseaux de télécommunication vers le "tout-IP", les modèles architecturaux au niveau réseau tiennent de plus en plus compte des notions de coopération entre les réseaux d'accès, de transport et de contrôle. Cette évolution au niveau réseau ne considère que le bout-en-bout transport. Elle coexiste avec des modèles architecturaux de niveau service, cependant, elle interagit d'une manière peu dynamique avec eux. Par conséquent, un passage de la coexistence à la cohabitation entre les couches réseaux et la couche service semble nécessaire. Cette cohabitation doit ainsi tenir compte de tous les défis qui résultent de l'évolution NGN, comme la mobilité, l'hétérogénéité et l'aspect *user-centric*. Au-delà de cette cohabitation entre les couches, une coopération au sein de la couche service semble aussi nécessaire. Par analogie à la convergence réseau, elle doit ainsi permettre de faire converger différentes caractéristiques appartenant à différents types de services (Telco, Web et IT) dans une même session de services. Cette session doit être personnalisée et doit tenir compte des préférences et de la QoS demandée par l'utilisateur.

La maîtrise de cette nouvelle génération de réseaux et de services (NGN/NGS) passe par la capacité des fournisseurs et des opérateurs à repenser leurs services de façon à être de plus en plus composables, et de les structurer ensuite dans une architecture convergente, autonome, distribuée et trans-organisationnelle. Le but est de lever les deux premiers verrous évoqués précédemment : concevoir un nouveau modèle de services qui répond à nos attentes, concevoir une nouvelle architecture de services qui assure la convergence et la personnalisation de services, et enfin concevoir les fonctionnalités qui permettent à cette architecture de cohabiter avec la couche NGN sous-jacente et d'homogénéiser l'environnement technologique des utilisateurs et des fournisseurs.

Nous commençons cette partie par une analyse des architectures de services existantes au niveau Telco, Web et IT (voir Chapitre 3). Nous les avons décrites et ensuite, nous avons discuté leurs avantages et leurs inconvénients afin de tirer des leçons à retenir pour appréhender ces nouveaux besoins qui régissent le nouveau contexte technologique NGN/NGS et *user-centric*. Par la suite, nous proposons dans le Chapitre 4 notre nouvelle architecture de services. Cette dernière est basée sur un modèle architectural (voir

Section 4.1) à base de *Middleware* qui supporte une approche évènementielle, orientée service, distribuée, horizontale, et qui s'appuie sur un nouveau modèle de services composables et à couplage lâches. Ensuite, nous proposons dans la Section 4.2 la nouvelle architecture de services notée NGN/NGS Middleware qui va garantir la composition convergente et personnalisée de services tout en tenant compte des défis provenant de l'émergence NGN. À la fin, nous concluons cette partie (voir Chapitre 5).

3 Analyse des architectures de services existantes

Dans le but de modéliser la couche applicative, différentes architectures sont proposées. Ces architectures existantes sont classées sous trois approches : Telco (voir Section 3.1), Web (voir Section 3.2) et IT (voir Section 3.3). À la fin, nous concluons ce chapitre (voir Section 3.4) en présentant un tableau qui récapitule et synthétise l'ensemble des valeurs ajoutées et des défis de ces différentes approches architecturales existantes.

3.1 Approche Telco

Cette approche a pour origine des opérateurs et des fournisseurs de services télécoms. Suite à l'évolution du NGN, elle se manifeste par une couche applicative indépendante des réseaux de contrôle, de transport et d'accès sous-jacents. Ces derniers fournissent des interfaces d'accès aux applications. Ces interfaces peuvent être des protocoles de signalisation comme dans le cas d'INAP (*Intelligent Networks Application Part*) dans les réseaux intelligents ou SIP dans l'interaction entre les serveurs d'applications SIP et le SCIM (*Service Capability Interaction Manager*). Les interfaces peuvent être aussi des APIs (*Application Programming Interfaces*) comme dans le cas de l'architecture Parlay/OSA (*Open Service Architecture*) et Parlay X. Dans le cas d'OMA (*Open Mobile Alliance*), l'interface est un ensemble de spécifications appelées *enablers*. Dans la suite, nous discutons, les architectures Telco que nous venons de citer.

3.1.1 Réseaux Intelligents

La conception des Réseaux Intelligents (RI) était la première étape vers des réseaux riches en fonctionnalités. Les premiers standards sont apparus dans les années 90. L'objectif essentiel des RI était de séparer la commutation qui se fait au niveau des commutateurs réseaux du traitement de la logique d'appel qui s'exécute dans une plate-forme de services. Dans cette section, nous commençons par décrire les dimensions architecturale et fonctionnelle de cette approche (voir Sous-Section 3.1.1.1). Ensuite, nous

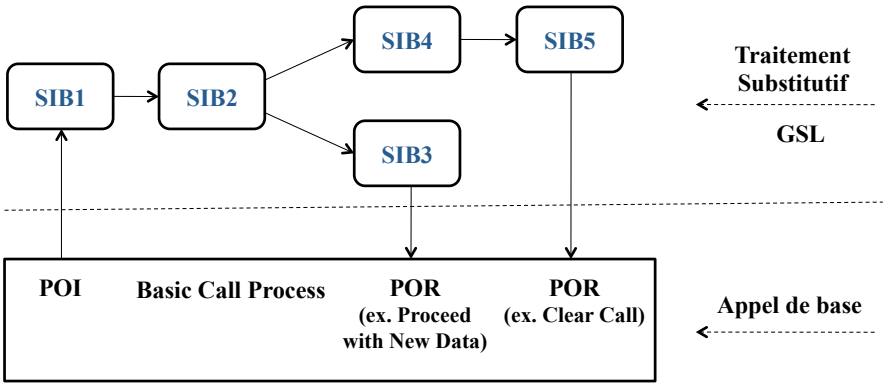

Fig. 3.1: Plan fonctionnel global des Réseaux Intelligents.

discutons ses principaux défis (voir Sous-Section 3.1.1.2).

3.1.1.1 Description architecturale et fonctionnelle des RI

Les RI ont été décrits selon plusieurs approches. La première considère le concept des RI comme un dispositif supplémentaire à l'architecture de commutation existante. Dans ce cas, les RI sont vus comme une collection de serveurs qui exécutent des fonctions spéciales de contrôle. La deuxième approche ne voit pas les RI comme un ensemble de nœuds ; elle les considère plutôt comme une architecture de logiciels qui exécutent des services. Afin d'homogénéiser les différents points de vue, l'ITU (*International Telecommunications Union*) standardise un modèle conceptuel, noté INCM (*Intelligent Network Conceptual Model*). Ce dernier réconcilie les approches physiques et logiques de l'architecture des RI. Pour ce faire, l'INCM est répati sur quatre plans de vision [28][40][52] : le plan de services, le plan fonctionnel global, le plan fonctionnel distribué et le plan physique.

Le plan de services est le plan supérieur de l'INCM. Il décrit, selon le point de vue de l'utilisateur, des services par un ensemble de caractéristiques, nommées *Service Elements Features*, sans préciser comment ces services sont implémentés dans le réseau. En effet, chaque caractéristique est vue comme une fonctionnalité réutilisable. L'idée clé des RI est donc de standardiser un ensemble de fonctionnalités (ex. *Call forwarding*, *Authentication*, *Call limiter*, etc.) qui seront réutilisées par les opérateurs de réseaux afin de composer leurs services.

Le deuxième plan dans l'INCM est le plan fonctionnel global. Il s'interface, au niveau architectural, juste au dessous du plan de services afin d'ajouter la vision des fournisseurs de services. En effet, le plan fonctionnel global décrit le service par une séquence de composants réutilisables, nommés SIBs (*Service Independent Building Blocks*). Ces derniers ne sont pas des programmes, mais plutôt des descriptions formelles d'activités agissant sur des données (ex. le numéro appelé, la *Black list*, etc.). Cette séquence de SIBs obéit à la logique globale de service, notée GSL (*Global Service Logic*), et représente le traitement substitutif exécuté par le RI par rapport à l'appel normal, noté BCP (*Basic Call Process*). Pour atteindre ce but, chaque SIB est muni d'un point logique d'initialisation, noté POI (*Point of Initialization*), et de plusieurs points logiques de

retour, notés POR (*Points of Return*). Par conséquent, comme le montre la Figure 3.1, lors d'un processus d'appel normal, BCP est interrompu par un processus substitutif. Ce dernier est déclenché par un POI (ex. Adresse analysée, Pas de réponse, etc.), puis exécuté par une séquence de SIBs selon une logique GSL bien déterminée. Une fois ce processus terminé, le service rend le contrôle au BCP. Le retour à l'appel normal pourra se faire par plusieurs types de POR (ex. Termine l'appel, Continue avec de nouvelles données, etc.). D'après les standards des RI, il y a 20 SIBs qui sont définis : *algorithm, authenticate, charge, compare, distribution, log call information, queue, screen, service data management, service filter, translate, user interaction, verify, BCP, BCUP, split, join, initiate service process, end, message handler.*

D'après la standardisation CS-2 (*Capability Sets - 2*), le processus de service substitutif n'est plus limité à une seule séquence de SIBs. En effet, lors de l'exécution d'un service RI, plusieurs processus peuvent être déclenchés en parallèle et peuvent ainsi communiquer par des messages. De plus, CS-2 améliore le concept de SIB en définissant un modèle récursif pour la création de services. Ce modèle classifie les SIBs sur différents niveaux de granularité. Pour ce faire, il différencie une opération SIB d'un SIB de haut niveau. La première représente une opération de base qui ne peut pas être décomposée en d'autres composants. Quand au SIB de haut niveau, noté HLSIB (*High-Level SIB*), il est décomposable en un ensemble d'opérations SIB, des SIBs normaux ainsi que d'autres HLSIBs.

Le troisième plan selon l'INCM est le plan fonctionnel réparti. Il s'interface, au niveau architectural, juste au dessous du plan fonctionnel global, afin de donner une vision plus réelle du réseau. En effet, le plan fonctionnel réparti définit un ensemble d'entités fonctionnelles qui servent à l'exécution de tout type de service défini en conformité avec les méthodes spécifiées dans les plans supérieurs. Le plan fonctionnel décompose les entités fonctionnelles en deux parties : celles qui permettent l'exécution des services RI et celles qui assurent la création et la gestion des services RI.

Nous citons ci-dessous quelques fonctions relatives à l'exécution des services RI :

- CCAF (*Call Control Agent Function*) : elle représente l'accès d'un utilisateur aux processus d'appel et de service. Elle supporte l'accès à des interfaces existantes comme les lignes téléphoniques analogiques, l'appel vers un réseau numérique à intégration de services, etc.
- CCF (*Call Control Function*) : elle contrôle l'état d'un appel. Par exemple, elle lance une alerte pour une ligne occupée.
- SSF (*Service Switching Function*) : Elle vérifie si les critères de débranchement vers un service substitutif sont vérifiés. Elle initialise ainsi la logique globale de services et interprète les commandes de la SCF.
- SCF (*Service Control Function*) : elle exécute le traitement substitutif et contrôle sa logique de services.
- SDF (*Service Data Function*) : elle représente la base de données que doit utilisée la SCF en temps réel afin d'exécuter un service RI.
- SRF (*Specialized Resource Function*) : elle contrôle des ressources spécialisées (essentiellement des serveurs vocaux) nécessaires à l'exécution des services RI.

Nous citons ci-dessous quelques fonctions relatives à la gestion des services RI :

- SMF (*Service Management Function*) : elle assure toutes les fonctions de gestion des services, comme l'installation des services RI, ajouter ou supprimer des abonnés, etc.
- SMAF (*Service Management Access Function*) : elle assure une interface d'accès (ex. Internet) entre le personnel chargé de la gestion de services et la SMF.
- SCEF (*Service Creation Environment Function*) : elle permet de définir, créer et tester des services RI puis de les transférer vers la SMF.

Le dernier plan de l'INCM est le plan physique. Il permet d'allouer des machines physiques pour les différentes entités fonctionnelles du plan fonctionnel réparti. Le plan physique correspond donc à l'architecture matérielle d'un réseau structuré en RI.

3.1.1.2 Défis des RI

La technologie réseau intelligent était la première tentative à introduire la notion de services. Pourtant, elle n'a pas pu poursuivre son avancement pour plusieurs raisons. Premièrement, cette technologie a rendu intelligent le processus d'appel, mais elle a eu du mal à s'exprimer dans des nouveaux environnements où la compréhension et la maîtrise des nouveaux concepts sont primordiales. Par exemple, la communauté des développeurs a mal interagit avec cet avancement technologique. Le deuxième problème des RI correspond à l'utilisation non adéquate du SIB. En effet, un SIB, selon le plan fonctionnel global, est un automate qui exécute une opération logique dans la logique de services. Donc, c'est l'opérateur et non pas l'opérande (la donnée) dans cette opération. En outre, selon son nom qui porte à confusion, les *Service Independent Building Blocks* ne sont pas des "Blocks" au sens composants de services, mais plutôt des descriptifs qui aident à la construction ("Building") des blocks de services. Vu que la majorité des industriels ont mal compris ce principe des SIBs, ils ont mis en cause la réutilisation d'un même SIB dans plusieurs services différents et ils ont considéré ainsi que le nombre de SIBs normalisés était limité. Pour cela, chacun a commencé à définir ces propres SIBs ce qui a mené au bout d'un an à réaliser 90 SIBs. Cette liberté de création non normalisée a abouti à des implémentations RI propriétaires qui nécessitent un coût et un effort très élevés pour garantir leur interopérabilité. Enfin, à l'aide des SIBs, les RI ont pu décrire l'enchaînement logique et ils l'ont associé à un ensemble de services au niveau applicatif. Cependant, la mauvaise définition des éléments de services au niveau du plan service, et la manque de normalisation des règles de composition de services et de structure ont causé des problèmes d'interaction et de portabilité des composants, et par conséquent des nouvelles défaillances pour l'approche RI.

3.1.2 Parlay/OSA

Une deuxième approche Telco, nommée Parlay/OSA (*Open Service Access*), a été créée suite à un travail commun réalisé par trois organisations : le 3GPP (*The Third Generation Partnership Program*), l'ETSI (*European Telecommunication Standards Institute*) et le groupe Parlay. Ce dernier est un consortium de multi-vendeurs ayant différents profils (fournisseurs de services, développeurs de logiciels, fournisseurs d'appareils réseaux, etc.). Dans cette section, nous commençons par décrire les dimensions architecturale et

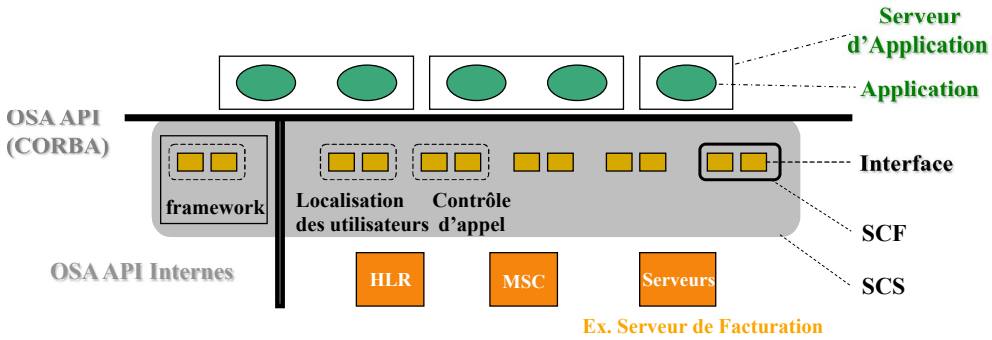

Fig. 3.2: Architecture du Parlay/OSA.

fonctionnelle de cette approche (voir Sous-Section 3.1.2.1). Ensuite, nous discutons ses principaux défis (voir Sous-Section 3.1.2.2).

3.1.2.1 Description architecturale et fonctionnelle du Parlay/OSA

Parlay/OSA [47][53] est une architecture de services ouverts, basée sur un ensemble d'APIs ouvertes, sécurisées et standardisées. Ces APIs sont facilement utilisables, riches en terme de fonctionnalités, et tiennent compte de l'innovation dans les entreprises. De plus, elles sont indépendantes des technologies du réseau sous-jacent (CDMA, GSM, etc.), ce qui permet ainsi le développement de nouvelles applications multi-réseaux et multi-environnements, et qui garantit par la suite la portabilité de ces applications sur des réseaux hétérogènes. Cette approche est aussi avantageuse par rapport aux opérateurs de réseaux qui peuvent ainsi introduire de nouvelles fonctionnalités sans avoir besoin de modifier les applications existantes.

Au niveau architectural, le modèle présenté dans la Figure 3.2 montre la médiation qu'assure la plate-forme Parlay/OSA entre les réseaux de télécommunications et les applications des opérateurs, des fournisseurs de services, des fournisseurs tiers et des MVNOs (*Mobile Virtual Network Operators*). Pour ce faire, elle se base sur des APIs (ex. *Call Control*, *User Interaction*, *Mobility APIs*, *Charging*, etc.) définies en CORBA (*Common Object Request Broker Architecture*). D'après ce modèle architectural, nous constatons le rôle central que joue le Parlay/OSA SCS (*Service Capability Server*), qui est une passerelle qui permet aux fournisseurs de services, aux fournisseurs tiers et aux MVNOs d'accéder aux applications internes offertes par l'opérateur propriétaire de ce SCS.

Au niveau fonctionnel, le modèle présenté dans la Figure 3.2 montre l'ensemble d'entités fonctionnelles qui viennent s'interfacer entre les applications fournies par les fournisseurs d'applications, et les services réseaux (commutation, facturation, etc.) fournis par les opérateurs. L'architecture OSA est composée essentiellement de deux entités fonctionnelles :

- SCF (*Service Capability Function*) : elle représente un ensemble d'interfaces CORBA standardisées qui correspondent à des classes d'opérations assurant l'accès des applications aux fonctionnalités du réseau sous-jacent. Il existe deux types de SCFs : le *framework* et les services réseaux. Le *Framework* fournit un ensemble

de fonctionnalités (ex. la gestion d'abonnement, l'enregistrement aux fonctionnalités réseaux et la sécurité d'accès) qui permettent un accès personnalisé et sécurisé aux services réseaux. Les services réseaux offrent l'accès aux fonctionnalités de réseau (ex. interaction de l'utilisateur, facturation, contrôle d'appel, etc.). D'après ce modèle architectural, nous pouvons aussi distinguer deux types d'APIs : les APIs internes, qui permettent l'interaction entre les différentes SCFs afin d'en tirer des nouvelles fonctionnalités et de nouveaux services à valeur ajoutée ; et les APIs externes, qui permettent l'accès des applications externes aux fonctionnalités SCFs d'un opérateur donné.

- SCS (*Service Capability Server*) : elle représente un serveur de fonctionnalités SCF. Il joue le rôle d'intermédiaire entre ses interfaces SCFs et les entités réseaux, en traduisant ainsi les invocations sur les SCFs en des évènements sur ces entités réseaux.

Afin de pouvoir interagir avec les autres entités fonctionnelles, le *framework* utilise trois interfaces d'interactions distinctes :

- Interface entre le *framework* et l'opérateur d'entreprise : elle permet l'abonnement à un service. En effet, l'opérateur d'entreprise joue le rôle d'un client qui souscrit à un service décrit par le *framework*. Une fois abonné, il autorise à ses applications de jouer le rôle d'utilisateur et de consommer ce service.
- Interface entre le *framework* et le SCS : elle permet l'enregistrement des SCFs. Chaque SCS contient un ensemble de SCFs qui doivent être publiées dans le *framework*. Ce dernier serait ainsi capable de décrire les SCFs disponibles lors d'une découverte lancée par des applications.
- Interface entre le *framework* et l'application : elle représente l'ensemble des mécanismes d'interactions entre les applications et l'architecture Parlay/OSA, permettant ainsi l'accès et l'usage des services OSA. Le cycle d'usage des services OSA est initié par une authentification suivie d'une autorisation de l'application cliente. Ensuite, l'application utilise l'interface de découverte afin de recevoir les interfaces *framework* disponibles, ainsi que celles des SCFs auxquelles elle a le droit d'accéder. Avant de pouvoir accéder aux SCFs, l'application doit signer la partie en-ligne du contrat de services, qui diffère de celle signée hors-ligne entre le *framework* et l'opérateur d'entreprise. Une fois que le contrat est signé, le *framework* permettra et contrôlera l'accès aux SCFs.

3.1.2.2 Défis du Parlay/OSA

L'approche Parlay/OSA a fait un grand travail pour assurer la portabilité des services auprès de différents réseaux hétérogènes. Cependant, elle ne peut pas être directement appliquée comme un moyen pour la gestion de la composition de services. En effet, elle n'a pas pu spécifier comment réutiliser et partager les blocs communs de composition de services. Les spécifications OSA couvrent seulement les fonctionnalités de contrôle d'accès aux services et les notions de découverte de ces services. En outre, l'approche Parlay/OSA permet une utilisation personnalisée des services par les applications clientes. Cependant, cette personnalisation se fait au moment de la procé-

dure d'abonnement et de signature de contrat ; ce qui rend impossible toute interaction temps réel avec l'utilisateur et par conséquent toute dynamique dans la composition de services.

D'autre part, l'architecture Parlay/OSA s'appuie sur une passerelle SCS qui utilise des OSA APIs afin de relier les applications clientes aux services réseaux fournis par l'opérateur. Deux inconvénients primordiaux découlent de cette approche architecturale. Le premier inconvénient est l'aspect monolithique qu'adopte cette architecture ainsi que sa forte dépendance des spécifications fonctionnelles et des APIs définies par Parlay/OSA. Cette architecture verticale de services suit une approche client/serveur qui se base sur les OSA APIs, ce qui empêche ainsi toute dynamique et transparence dans la composition de services hétérogènes, et qui augmente le nombre d'interactions et le temps nécessaire pour obtenir une session complète de services. Le deuxième inconvénient est la centralisation d'accès aux services. En effet, toutes les requêtes applicatives doivent passer obligatoirement par le SCS pour atteindre les services, ce qui cause ainsi un étranglement au niveau du SCS qui devient un point de défaillance, et par conséquent une dégradation des performances.

Afin d'assurer une plus grande souplesse et un plus haut niveau d'abstraction aux développeurs, Parlay a conçu une nouvelle approche, nommée Parlay X [32]. Cette architecture propose un ensemble d'*API Web Services* qui sont sélectionnés selon une utilité commerciale et non nécessairement selon une élégance technique. Le but est de définir un ensemble de possibilités et de capacités puissantes pourtant simples, fortement abstraites, imaginatives, de télécommunications que les développeurs puissent rapidement comprendre et utiliser afin de créer de nouvelles applications innovantes. Au niveau du modèle architectural, le *Parlay X Gateway* est utilisé comme une passerelle qui permet aux fournisseurs de services, aux MVNOs et aux fournisseurs tiers d'accéder par une interface Web aux services internes offerts par l'opérateur propriétaire de cette passerelle. Malgré l'ouverture que propose ces *API Web Services*, l'architecture Parlay n'a pas pu dépasser ses défis, en conservant toujours une approche verticale client/serveur.

3.1.3 OMA

L'OMA (*Open Mobile Alliance*) est une organisation internationale, constituée en Juin 2002 par un ensemble de fournisseurs d'applications, des entreprises du monde IT, et des opérateurs mobiles. Son objectif est de développer des spécifications ouvertes et orientées usagers pour des services mobiles interopérables. Ces derniers sont conçus pour être indépendants des régions géographiques, des opérateurs mobiles multiples, des terminaux et des systèmes d'exploitation. En effet, OMA cherche à garantir à tout utilisateur la possibilité d'accéder et d'échanger des informations en utilisant n'importe quels terminal, service ou réseau. Dans cette section, nous commençons par décrire les dimensions architecturale et fonctionnelle de cette approche (voir Sous-Section 3.1.3.1). Ensuite, nous discutons ses principaux défis (voir Sous-Section 3.1.3.2).

3.1.3.1 Description architecturale et fonctionnelle d'OMA

Au niveau architectural, OMA [48][9] cherche à proposer une architecture qui assure l'intégrité, le passage à l'échelle, l'interopérabilité, la sécurité, et surtout l'élimination des approches monolithiques. Pour atteindre ces objectifs, OMA base son modèle architectural sur des spécifications appelées *enablers*. Ces derniers sont des blocks standardisés qui étaient conçus au début pour la composition des services mobiles. Mais, au cours des années, plusieurs spécifications liées aux réseaux fixes apparaissent, ce qui a conduit à la prolifération des services mobiles tout en permettant l'interopérabilité entre les réseaux fixes et mobiles. Il faut bien noter que les spécifications devraient préciser comment se fait l'invocation des *enablers* et l'appel de leurs fonctions. De plus, toutes les exigences et les fonctions qui ne sont pas intrinsèques à un *enabler* ne devraient pas être définies dans sa spécification. La spécification d'un *enabler* devrait seulement spécifier la fonctionnalité intrinsèque exigée pour accomplir son rôle.

En effet, l'utilisation de ces *enablers* est très avantageuse. Tout d'abord, ils représentent des composants interopérables qui permettent l'interaction entre différents composants et applications développés par des fournisseurs hétérogènes. En outre, les spécifications des *enablers* réduisent les efforts de déploiement et permettent aux mêmes applications de fonctionner à travers une large variété d'environnements d'une façon cohérente. De plus, OMA tient compte du principe de réutilisation dans ses spécifications, ce qui lui permet de réduire le problème des architectures verticales en-silos. En effet, chaque spécification d'*enabler* peut réutiliser d'autres spécifications existantes. Par exemple, l'*enabler Push to talk over Cellular* réutilise les *enablers* de présence et de gestion de groupe.

Ayant décrit ces *enablers*, la question qui se pose est comment les structurer, les intégrer et les gérer ? Pour répondre à cette question, OMA a proposé un environnement d'intégration unifiée pour ses *enablers*, et l'a nommé OSE (*OMA Service Environment*) [48][9]. Ce dernier assure une ouverture des *enablers* vers des nouveaux développeurs et fournisseurs tiers afin de permettre le développement et le déploiement de nouvelles applications innovantes. Cependant, il faut noter qu'un des principes fondamentaux d'OMA consiste à ne pas spécifier des APIs. De plus, OMA ne précise pas où se trouvent les éléments architecturaux (applications, *enablers*, etc.). Ils peuvent appartenir à différents domaines OSE (domaine de l'opérateur mobile, domaine du terminal, etc.) et peuvent ainsi interagir indépendamment de leurs localisations. En outre, dans n'importe quel domaine OSE, les implémentations des *enablers* exposent des interfaces standards pour l'application et l'usage des *enablers*. Ces implémentations se relient aux ressources actuelles présentes dans le domaine OSE. A l'aide de cette abstraction, il sera possible d'ajouter ou de modifier les ressources sous-jacentes sans affecter l'interface exposée par les implémentations des *enablers*, et sans affecter ainsi les applications, ce qui est très important dans le cas de différents vendeurs, technologies réseaux et fournisseurs.

D'après la Figure 3.3, nous pouvons distinguer les différents éléments constituant l'architecture OSE :

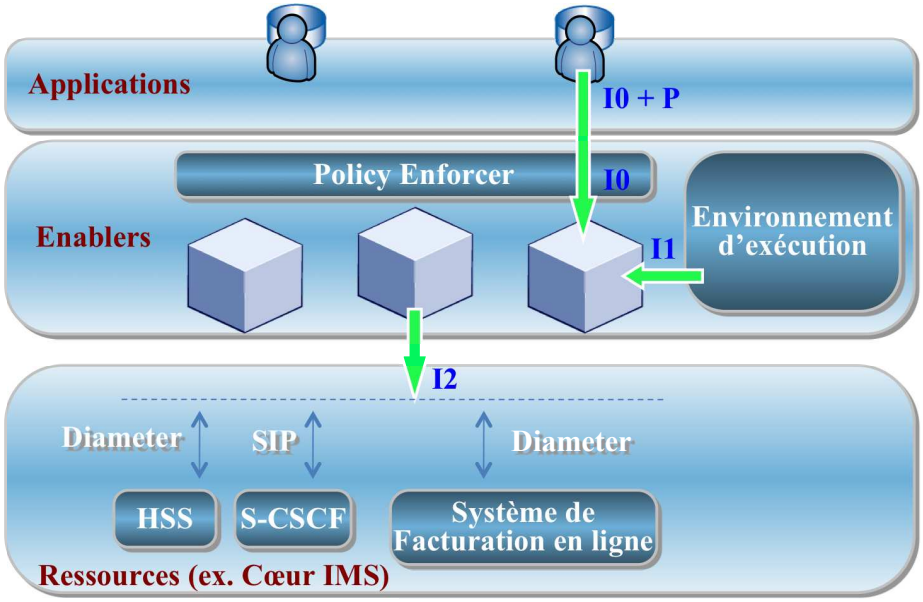

Fig. 3.3: Architecture de l'*OMA Service Environment*.

- *Enablers* : ce sont des composants définis dans des spécifications et qui sont réutilisés afin de faciliter le développement, le déploiement et l'exploitation des nouveaux services.
- Les interfaces : l'interface *I0* permet l'accès aux fonctions intrinsèques des *enablers* invoqués. En effet, l'interface d'un *enabler* peut être enregistrée auprès d'un *enabler* de découverte afin de permettre aux applications de s'attacher dynamiquement avec l'*enabler* de destination. L'interface *I1* permet aux *enablers* de s'exécuter dans un environnement d'exécution. L'interface *I2* permet aux *enablers* d'accéder aux ressources du réseau sous-jacent. Enfin, l'interface *I0+P* permet l'imposition de nouveaux paramètres "*P*" (ex. personnalisation de l'usager, autorisation d'accès, etc.) qui viennent s'ajouter aux politiques de gestion déjà exécutées. Ces paramètres doivent être fournis dans la requête envoyée vers l'interface *I0* d'un *enabler*. Normalement, c'est l'administrateur du domaine de l'OSE contenant l'*enabler* en question qui impose ces paramètres.
- *Enabler interface bindings* : les interfaces doivent être spécifiées selon un langage neutre. Cependant, les spécifications peuvent aussi définir des *bindings* pour ces interfaces. Les *enabler interface bindings* fournissent ainsi des formats spécifiques pour l'accès aux *enablers*, tout en utilisant des langages particuliers de programmation (ex. Java ou C) ou bien des services web.
- Environnement d'exécution : c'est une plate-forme qui englobe plusieurs fonctions comme la surveillance du processus, gestion du cycle de vie de ce processus, l'équilibrage de charges (*Load Balancing*), ainsi que d'autres fonctions de gestion et d'administration. Ces fonctions peuvent ne pas être directement exposables pour les applications, mais elles peuvent être directement invoquées par les implémentations des *enablers*.
- *Policy enforcer* : il assure un mécanisme de gestion par politique. Il permet de

protéger les *enablers* des requêtes non autorisées et d'évaluer puis appliquer les préférences de l'utilisateur.
- Ressources : elles représentent l'ensemble des capacités que fournit le domaine sous-jacent, que cela soit le réseau du fournisseur de services, celui de l'opérateur ou celui du terminal. Dans l'OSE, l'implémentation d'un *enabler* peut directement invoquer ou accéder aux ressources.
- Application : c'est une implémentation d'un ensemble de fonctions qui sont reliées afin d'assurer un travail bien défini. Elles sont des moyens primordiaux pour initier et utiliser un *enabler*. Il faut noter que ces applications peuvent être internes ou externes à l'OSE contenant les *enablers* en question.

3.1.3.2 Défis d'OMA

L'architecture OMA a pu dépasser les problèmes des architectures Telco existantes. Elle a remplacé l'approche monolithique par un ensemble d'*enablers* interopérables qui sont séparés des ressources qu'ils utilisent, ainsi que des applications qui les utilisent. Cependant, cette séparation n'assure pas en elle-même la réutilisation des *enablers*. En effet, cette caractéristique nécessite une exposition des fonctions fournies par ces *enablers* sur des interfaces bien définies. Pourtant, des considérations doivent être prises auprès de ces interfaces afin de permettre non seulement des interopérations prédéfinies mais aussi des réutilisations inhabituelles. Un deuxième défi que nous pouvons souligner et qui limite la prolifération de l'architecture OMA au sein des entreprises IT, des fournisseurs de services et des opérateurs, est la limitation de cette approche aux spécifications des *enablers* OMA. Cela empêche ainsi toute possibilité de coopération avec des architectures existantes et toute ouverture possible vers des nouvelles approches.

De plus, au cours des années de standardisation d'OMA, il y avait une coexistence entre deux philosophies de standardisation différentes. La première est basée sur une architecture globale qui essaye de garder la cohérence et la consistance entre les spécifications d'OMA, tout en les structurant dans un cadre architectural bien défini. La deuxième approche est celle des sociétés qui développent certaines fonctionnalités pour être réutilisées par celui qui les trouve appropriées, mais sans se soucier des superpositions ou du manque d'intégration dans l'environnement global. Cette superposition ne se limite pas au niveau des fonctions spécifiées par OMA, mais elle dérive vers une divergence dans l'intégration avec les autres systèmes. Par exemple, différents *enablers* dans différents domaines peuvent avoir différentes approches pour gérer l'abonnement de l'utilisateur, sa confidentialité, ses informations, la facturation, etc. Enfin, à cause de la coexistence des ces philosophies, un grand effort doit être fait au sein d'OMA afin d'éviter toute superposition possible et de créer ainsi une certaine synergie. Ainsi, malgré le travail de promotion pour réutiliser les *enablers* et éviter les silos, un travail d'optimisation et de communication locale semble aussi nécessaire.

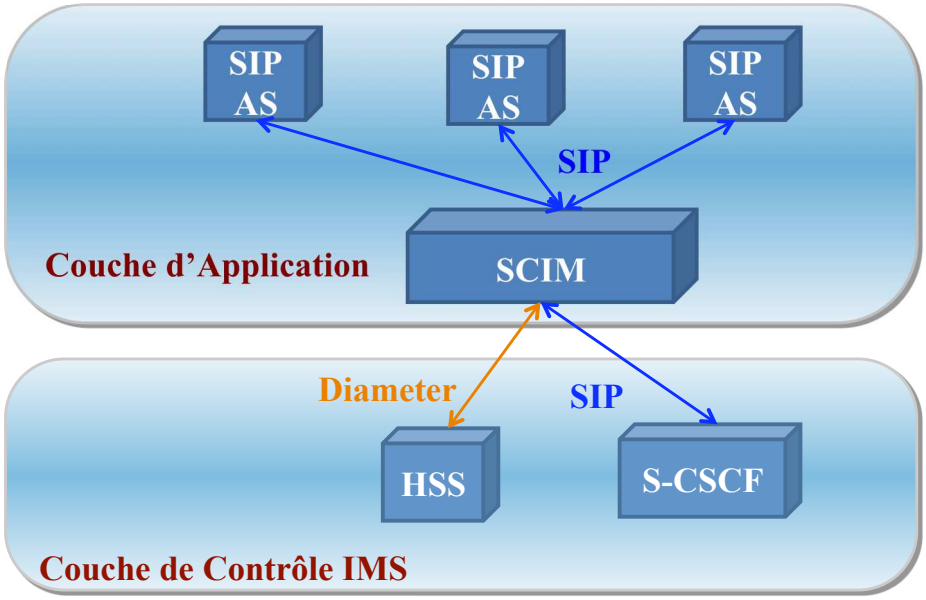

Fig. 3.4: Architecture de l'IMS-SCIM.

3.1.4 IMS-SCIM

Avec l'avènement des réseaux de nouvelle génération NGN, une couche de contrôle IMS (*IP Multimedia Subsystem*) avait été conçue par le 3GPP afin de garantir l'établissement, la maintenance et la fermeture de la session multimédia, et afin de contrôler cette session de bout-en-bout tout en utilisant le protocole de signalisation SIP. Dans cette section, nous commençons par décrire l'approche architecturale et fonctionnelle proposée par IMS afin de supporter l'accès et l'usage des services multimédias (voir Sous-Section 3.1.4.1). Ensuite, nous discutons ses principaux défis (voir Sous-Section 3.1.4.2).

3.1.4.1 Description architecturale et fonctionnelle d'IMS-SCIM

Au niveau architectural, IMS [2] représente une couche de contrôle introduite entre les couches de transport et la couche de services. Il assure ainsi un cadre unifié de prestations de services pour les réseaux fixes et mobiles. Afin d'invoquer et de contrôler ces services, IMS utilise le S-CSCF (*Serving Call Session Control Function*) avec son interface ISC (*IP Multimedia Service Control*) qui est basée sur SIP. Dans ses standards IMS, le 3GPP représente la couche service par un ensemble de blocs constructifs nommés *Service Capabilities* (SC). Ces derniers sont définis en tant que fonctionnalités autonomes qui peuvent être réutilisées par différents serveurs d'applications. Par exemple, la présence, la messagerie instantanée, la conférence et la messagerie multimédia sont des SCs qui sont réutilisées dans différentes applications comme les jeux multimédias, les forums de discussion en ligne, le *e-learning*, etc. Cette approche de services a pour objectif de permettre aux fournisseurs d'applications de créer et de gérer leurs nouvelles applications en se servant des SCs déjà existantes. De plus, elle cherche à éviter toute redondance au niveau de ces fonctionnalités utilisées.

Puisque plusieurs SCs peuvent être invoquées pour l'exécution d'une seule application, et puisque plusieurs de ces SCs peuvent être utilisées par plusieurs applications, des nouvelles fonctionnalités semblent ainsi nécessaires au niveau de l'architecture de services, afin de gérer les invocations multiples et simultanées de SCs. Pour atteindre ce but, 3GPP implémente ces fonctionnalités exigées dans un serveur SIP d'applications, nommé SCIM (*Service Capability Interaction Manager*) [31]. Ce dernier est une entité fonctionnelle qui gère l'interopérabilité et la coopération entre les différents serveurs d'applications et qui contrôle comment les SCs sont réutilisées par différentes applications. Au niveau architectural, comme le montre la Figure 3.4, le SCIM se manifeste comme une passerelle entre le S-CSCF et les serveurs d'applications. De plus, SCIM représente aussi une frontière entre le domaine de l'opérateur du réseau et le domaine des fournisseurs de services. Ainsi, en incluant des mécanismes de gestion de la composition de services dans SCIM, différents fournisseurs de services pourraient introduire des serveurs d'applications à valeurs ajoutées tout en réutilisant un ensemble de SCs fournies par l'opérateur du réseau IMS. Ensuite, SCIM permet à l'opérateur du réseau IMS de contrôler la coopération et l'interopérabilité entre ces serveurs en se basant sur les contraintes, les préférences et les accords spécifiés entre l'opérateur du réseau et les fournisseurs de services.

3.1.4.2 Défis d'IMS-SCIM

Malgré l'avancement assuré par SCIM au niveau de l'architecture IMS, ce serveur SIP d'applications n'est pas encore standardisé. Cela a mené ainsi à concevoir ce SCIM selon différentes approches et à lui attribuer différentes définitions : certains le considèrent comme un *dispatcher* des logiques de services dans un environnement local d'exécution ; d'autres le définissent comme étant une entité qui gère l'interaction entre les composants qui implémentent les serveurs proxy SIP ou les agents utilisateurs ; et d'autres encore le considèrent comme étant une entité qui gère l'interaction entre les fonctions SIP et les composants du système de signalisation. En effet, le choix parmi ces différents comportements va dépendre essentiellement du modèle technologique et économique adopté par les différents fournisseurs de services et opérateurs IMS.

Dû à ce manque de standardisation, non seulement la définition du SCIM n'est pas claire, mais les fonctionnalités et les mécanismes de gestion de l'interaction des services ne sont pas encore spécifiées. Par exemple, il n'y a aucune spécification qui montre comment le SCIM va coordonner plusieurs invocations de services.

En outre, l'approche architecturale du SCIM est fortement reliée à l'avancement d'IMS et à son adoption dans les réseaux de futur. De plus, SCIM est un composant central par lequel doivent passer toutes les requêtes d'invocation des services. Cela peut causer ainsi un problème d'étranglement au niveau du point de défaillance qui est le SCIM, et par conséquent une dégradation des performances.

3.2 Approche Web

Vers la fin des années 90, un nouveau besoin s'est fait sentir : celui de la communication entre différentes applications et l'interopérabilité entre des environnements hétérogènes tout en étant indépendant des technologies et des capacités réseaux sous-jacentes. Dès lors, les services Web sont apparus. Dans cette partie, nous allons définir les services Web (voir Sous-Section 3.2.1) et décrire ainsi tous les concepts qui lui sont reliés. Ensuite, nous présentons le concept Web 2.0 (voir Sous-Section 3.2.2). Enfin, nous discutons les principaux défis de cette approche Web (voir Sous-Section 3.2.3).

3.2.1 Services Web

Selon la définition du W3C (*World Wide Web Consortium*) [8], un service Web est considéré comme un logiciel système désigné pour supporter l'interopérabilité des interactions entre différentes machines dans le réseau. Il a une interface décrite normalement par un format WSDL (*Web Service Description Language*). D'autres systèmes interagissent avec le service Web d'une manière précisée dans sa description et en se basant sur des messages SOAP (*Simple Object Access Protocol*) qui sont normalement transportés en utilisant des protocoles comme HTTP (*Hypertext Transfer Protocol*), avec une sérialisation XML (*eXtensible Markup Language*) et avec la contribution d'autres standards reliés à l'approche Web. En effet, la logique d'implémentation d'un service Web peut être écrite dans n'importe quel langage et dans n'importe quelle plate-forme, et elle est accessible par un ensemble de technologies Web standardisées. Ceci permet ainsi un accès plus rapide aux services Web déployés et une communication plus efficace entre les applications concernées.

Au niveau architectural, un service Web est représenté comme un composant appli-

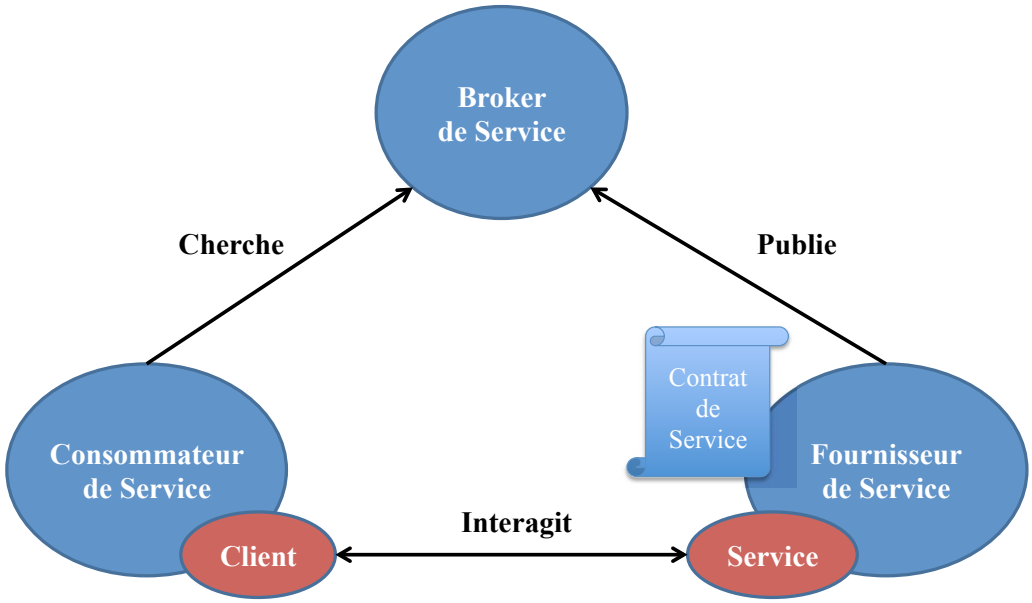

Fig. 3.5: Architecture des services Web.

catif composable à couplage lâche. L'architecture des services Web, comme le montre la Figure 3.5, se base sur trois acteurs [8] :

- Le fournisseur de services : il fournit des services et les expose aux consommateurs en les publiant.
- Le courtier (*Service Broker*) : il joue le rôle d'un annuaire intermédiaire entre le fournisseur et le consommateur de services. Il permet au fournisseur de publier ces services Web selon des fichiers WSDL. Il aide aussi à la recherche d'un service Web, en précisant dans sa réponse la description WSDL de ce service, la description des données qui composent les requêtes vers ce service, ainsi que le protocole à utiliser dans cette invocation.
- Le consommateur de services : il représente un client qui cherche puis accède à un service Web. Il interroge le *Service Broker* pour trouver un service, puis invoque ce service en envoyant des requêtes au fournisseur correspondant.

Au niveau technologique, l'approche des services Web se base sur un ensemble de standards répartis sous plusieurs couches [8] : la couche de communications, la couche de messages, la couche de descriptions, la couche de processus, ainsi que les interfaces de gestion et de sécurité. Au niveau de la couche de communications, un ensemble de protocoles applicatifs sont définis pour assurer le transport des messages à travers le réseau Internet, comme par exemple, HTTP, SMTP, FTP, etc. Au niveau de la couche des messages, XML est utilisé dans les normes du W3C afin de définir un format de données standard, flexible et extensible qui garantie une dynamique dans la création et la gestion du contenu et qui permet de réduire nettement la charge de déploiement des technologies nécessaires au succès des services Web. L'approche XML est basée essentiellement sur les technologies suivantes [8] : le XSD (*XML Schema Definition*), qui décrit la structure d'un document XML et définit ses différents blocs constitutifs (les éléments, les attributs, les types de données, etc.) ; le *XML namespaces*, qui définit des noms uniques attribués aux différents éléments dans un document XML ; et le XSLT (*eXtensible Style-sheet Language Transformations*), qui est utilisé pour assurer la transformation des documents XML en d'autres formats de documents.

Afin de spécifier l'échange des messages XML dans une architecture de services Web, le W3C a adopté dans ses normes le protocole SOAP [33]. Ce dernier est conçu pour simplifier l'accès aux objets distants tout en étant indépendant de l'implémentation technique du service et de la technologie de transport sous-jacente. De plus, SOAP décrit les informations échangées selon une syntaxe qui correspond bien aux fonctions applicatives. Ces dernières sont représentées ainsi par un ensemble extensible de verbes. Afin d'identifier un message SOAP dans la syntaxe XML, nous utilisons le champ *SOAP Envelope* qui est décomposé en deux parties : le *SOAP Header* contenant des informations et des directives complémentaires pour le traitement des données (algorithme de chiffrement, identificateur de la source du message, position du message qui peut être soit un point intermédiaire soit final, etc.) ; et le *SOAP Body* contenant les données échangées qui suivent une structure de données spécifiques et une syntaxe propre à la fonction applicative en question.

SOAP n'est pas le seul protocole d'échange de messages dans le monde du Web. En effet, au cours de l'an 2000, *Roy Fielding* a conçu un nouveau protocole, nommé

REST (*REpresentational State Transfer*), afin de rivaliser le SOAP. Selon Fielding [24], SOAP est incapable de se comporter comme une partie du Web à cause de sa tendance à encapsuler toutes sortes d'actions sous forme d'une interface spécifique à un objet, ce qui rend l'approche de plus en plus complexe. En effet, REST [24] est un protocole plus simple que SOAP. Il représente le monde par un ensemble de ressources auxquelles il attribue des identifiants uniques, nommé URI (*Unified Resource Identifier*). Par conséquent, pour accéder à une ressource, il suffit que la requête porte lisiblement l'URL (*Unified Resource Locator*) correspondante. En outre, la simplicité de REST est dûe aussi à l'utilisation des méthodes de base de HTTP (*GET*, *PUT*, *POST* et *DELETE*) lors de la consultation ou de la mise à jour d'une ressource. Cependant, le nombre de verbes (méthodes du protocole applicatif) n'est plus extensible comme dans SOAP. Au niveau conceptuel, comme son nom l'indique, REST se base sur le principe de transition d'états. Pour cela, il représente l'architecture Web comme un réseau de ressources. Ensuite, il permet à l'utilisateur, lors de l'exécution de son application, d'utiliser des liens (transition d'états) pour passer d'une ressource (état de l'application) vers une autre.

Nous présentons ci-dessous la liste des différences entre SOAP et REST :

- REST adopte une approche de ressources qui sont identifiées par des URIs. Quand à SOAP, il adopte une approche de services dont chacun est représenté par un contrat et un ensemble de méthodes spécifiques.
- Dans REST, les clients passent d'un état à un autre en utilisant les URLs attribuées aux ressources. Dans SOAP, l'interaction se fait à l'aide d'un système de contrôle unifié, c.à.d. le bus de services ou bien le gestionnaire de messages SOAP. De plus, l'URL dans SOAP est masquée dans l'enveloppe, ce qui empêche toute aide possible de la part d'un serveur proxy ou d'un serveur de cache.
- L'interface d'invocation dans REST est standardisée et uniforme pour toutes les ressources. Elle utilise les méthodes de base de HTTP. Dans SOAP, l'interface dépend de la méthode invoquée qui est spécifique pour chaque service (ex. lireCatalogue, récupérerEtatDemande, etc.). Le but est d'avoir des contrats basés sur des verbes. Par conséquent, SOAP est plus extensible mais moins simple.
- REST adopte une interaction client/serveur et synchrone dans le Web. Quand à SOAP, il permet une composition de services synchrone ou asynchrone, une meilleure interopérabilité et plus de sémantique. Cependant, elle nécessite l'installation et la médiation de nouveaux logiciels comme par exemple les bus de services synchrones ou asynchrones.
- Au niveau du routage des messages dans REST, il est basé sur les URLs lisibles dans les messages. Quand à SOAP, il parte du principe que le client doit se renseigner sur le contrat des services et sur la sémantique de leurs interfaces afin de pouvoir les sollicités.
- REST est utilisé essentiellement par les développeurs Web et surtout dans le cadre du Web 2.0. Il répond à des besoins d'intégration simple au niveau de l'interface utilisateur, dans des services sur Internet (ex. services destinés aux utilisateurs afin de les permettre de partager des informations entre plusieurs ap-

plications). SOAP est plutôt utilisé dans des compositions de services complexes qui nécessitent de la sémantique.

Au niveau de la couche de description, le WSDL [8] est utilisé. Ce dernier est un langage XML qui décrit l'interface d'un service Web et son fonctionnement, afin de permettre son invocation et son interaction avec d'autres services Web à l'aide des messages SOAP. Le WSDL est nécessaire puisque les messages SOAP ne spécifient pas le format concret des entrées/sorties qu'un service Web est capable de recevoir, de traiter et de renvoyer. Il faut noter qu'il y a deux versions de WSDL. La première est la version WSDL1.1 [11] qui s'est contentée de décrire le service en terme statique sans aucun engagement au niveau de la qualité de services. Ensuite, la deuxième version, WSDL2.0 [10], a été conçue afin d'étendre la description du service Web en ajoutant des éléments de qualité de services.

La dernière couche de la pile technologique des services Web est celle des processus. Or, pour créer un processus de services Web, il faut d'abord commencer par la découverte des services Web impliqués afin de permettre par la suite leurs interactions. Pour faciliter la découverte des services Web, le W3C a standardisé en 2000 le principe d'UDDI (*Universal Description, Discovery, and Integration*). L'organisation OASIS (*Organization for the Advancement of Structured Information Standards*) a repris la gestion de ce principe et a créé la version 3 en 2004. En effet, UDDI [12] est vu comme un annuaire dans lequel les fournisseurs décrivent et publient leurs services, et à partir duquel les consommateurs découvrent les services demandés. Au niveau technologique, l'UDDI s'appuie sur WSDL pour la description des contrats de services hébergés chez lui, et sur des requêtes SOAP pour pourvoir y accéder. Afin de faciliter la découverte dans l'UDDI, l'annuaire est décomposé selon trois types d'informations : les pages blanches contenant des informations sur les fournisseurs de services, les pages jaunes contenant des informations sur les catégories des services, et les pages vertes contenant des informations sur les contrats WSDL. Des SOAP APIs sont utilisées par les fournisseurs afin de publier leurs services selon ces trois types de pages.

D'autre part, les annuaires UDDIs peuvent être distingués par leurs types d'accès : l'UDDI public qui est accessible par tous les fournisseurs et les consommateurs de services Web ; l'UDDI privé qui est utilisé dans des réseaux isolés pour un usage interne et sécurisé ; et l'UDDI protégé qui limite l'accès à certains collaborateurs faisant partie du cercle de confiance. Dans la pratique, les annuaires UDDI publics n'existent pas. D'autre part, suite à la conception des architectures fédérées entre plusieurs UDDI de différent partenaires, l'accès UDDI protégé devient de plus en plus déployé. Pour ce faire, UDDI essaye de normaliser la notion d'affiliation entre ces annuaires, en identifiant chaque service par une clé primaire et unique.

Après avoir présenté l'ensemble des technologies Web qui définissent le format des messages échangés entre les services, qui décrivent les interfaces de ces services et qui aident à leur découverte, nous présentons maintenant les deux approches qui assurent la composition de ces services Web de façon à créer des processus plus complexes :

– L'orchestration : elle correspond à une partie exécutable d'un processus de ser-

vices inter-organisationnel. Elle est exécutée par un outil d'orchestration. Ce processus exécutable est décrit par un modèle d'orchestration qui définit des actions de communications ainsi que des actions internes. Les actions de communication correspondent aux interactions avec des services Web externes provenant d'autres partenaires. Les actions internes correspondent à des interactions avec des services internes. Parmi les différents langages qui assurent l'orchestration lors d'une composition de services, BPEL (*Business Process Execution Language*) est devenu le standard le plus utilisé. Au niveau architectural, BPEL se représente sous deux modèles possibles [7] : centralisé et décentralisé. Dans le modèle centralisé, l'orchestrateur BPEL agit comme un point central de coordination pour gérer la collaboration entre les services Web. Il est ainsi considéré comme un service Web ayant une spécification WSDL qui dérive de sa spécification BPEL. Par conséquent, l'orchestrateur BPEL est responsable d'acheminer les messages SOAP entre les différents services constituant le processus demandé, tout en se basant sur la spécification BPEL du processus. D'après le modèle décentralisé, la communication pour orchestrer le processus est faite entre plusieurs orchestrateurs BPEL. Par conséquent, chaque processus BPEL, relatif à un partenaire, est exécuté en local par l'orchestrateur BPEL correspondant. Ensuite, chaque orchestrateur se présente à l'extérieur sous forme d'un service Web, et peut ainsi collaborer avec les autres orchestrateurs en échangeant des messages SOAP, ce qui permet ainsi la coordination des différentes exécutions locales pour constituer le processus global.

– La chorégraphie : elle correspond à des séquences de messages entre plusieurs services dans un processus inter-organisationnel. Donc, elle constitue un certain accord entre différentes parties sur des principes de collaboration. Cet accord est représenté par un modèle de chorégraphie qui décrit la collaboration entre l'ensemble des services dans le but de composer un processus à valeur ajoutée. Pour ce faire, le modèle procède à la saisie des interactions entre les services ainsi que celle des dépendances entre ces interactions. Nous citons par exemple les dépendances du flux de contrôle (ex. une interaction doit être faite avant une autre), les dépendances du flux de données, les corrélations de messages, les contraintes de temps, etc. En outre, une chorégraphie ne décrit aucune action interne aux services participants si elle ne donne aucun résultat visible à l'extérieur de ce service ; par exemple, la transformation de données, ou bien une action de calcul interne au service. Il y en a plusieurs langages qui assurent la chorégraphie lors d'une composition de services. WS-CDL (*Web Services Choreography Description Language*) est l'un de ces langages. Il est le standard recommandé par le W3C pour spécifier la chorégraphie de services. La version WS-CDL1.0 [42] est spécifiée en 2005 par W3C. WS-CDL est un langage déclaratif basé sur XML, qui a pour objectif de spécifier les protocoles de collaboration entre différents services Web. Il fournit une vision publique globale sur l'ensemble des participants qui coopèrent, selon un modèle *Peer-To-Peer* (P2P), en fournissant des services Web distribués afin d'atteindre un objectif métier commun. WS-CDL décrit leurs

comportements à l'aide de l'ordre des messages qu'ils échangent ainsi que les opérations qu'ils fournissent.

3.2.2 Web 2.0

L'approche des services Web a causé un changement fondamental dans la manière de concevoir et réaliser les applications informatiques et de programmer les ordinateurs. De même, elle a fait un progrès vers une interaction de plus en plus dynamique de la part des internautes dans le monde du Web. Ce progrès prend de l'ampleur avec l'apparition du concept de Web 2.0 [6]. Ce dernier permet à l'utilisateur de partager de l'information sur le Web, en utilisant les réseaux sociaux et les outils collaboratifs de création et de partage d'informations, comme les blogs, les wikis, etc. Cette approche est ainsi une vraie avancée technologique par rapport à la version Web 1.0 qui représente l'utilisateur comme un simple navigateur et lecteur d'informations. Cependant, cette nouvelle vision du Web qui permet de maintenir des connexions plus fluides, plus flexibles et plus riches entre les utilisateurs et le monde du Web, doit être accompagnée par une progression technique surtout aux niveaux de l'accès et de l'interaction dynamique, d'où l'émergence des nouvelles techniques comme les *Mashups*, les composants AJAX (*Asynchronous JavaScript and XML*) et les services REST.

L'approche Web 2.0 s'intéresse d'abord à l'accès, partage et fourniture de données. Pour cela, elle se base sur une approche architecturale REST qui représente le Web par un ensemble de ressources accédées simplement par les services basiques d'accès, nommés CRUD (*Create, Read, Update and Delete*). Pour plus d'efficacité, le Web 2.0 s'appuie sur des *Mashups* [56] afin de construire le plus vite possible des applications exploitant ces services CRUD. En effet, le *Mashup* est une application composite publiée et accessible sur le Web, permettant de combiner plusieurs services fournis par des sites Web, afin de créer une application finale. Il faut noter que le *Mashup* invoque les services en utilisant l'approche REST, c.à.d. en utilisant les méthodes de base de HTTP. De plus, ils s'appuient sur *JavaScript* et XML.

Cependant, Web 2.0 ne pouvait se contenter ni du dialogue Web classique, celui du Web 1.0, consistant à recharger toute la page Web lors de chaque interaction de l'utilisateur, ni des composants graphiques limités. Pour cette raison, la technique AJAX [56] est créée et adoptée par le Web 2.0. Au niveau conceptuel, AJAX assure plus d'ergonomie aux utilisateurs du Web 2.0. Elle permet d'offrir des composants graphiques plus évolués que ceux offerts par HTML, et d'empêcher le rechargement d'une page HTML lors de chaque action de la part d'un utilisateur. De plus, AJAX s'appuie toujours sur des techniques Web de base, comme HTML, HTTP et *JavaScript*.

3.2.3 Défis de l'architecture Web

L'architecture Web a fait un grand pas en avant au niveau de la sémantique de création, de collaboration et de réutilisation des services. Cette architecture fortement distribuée se base sur des *Web Services APIs* qui sont nettement plus intelligentes que les APIs proposées par les architectures Telco. Malgré ses avancements, l'approche Web

est encore confrontée à des défis au niveau de l'invocation, de la découverte et de la composition des services.

Au niveau de l'invocation des services, l'architecture Web suit toujours une approche client/serveur qui est basée sur l'utilisation des *Web Services APIs*. Cette approche existait dans le Web 1.0, mais prend plus de l'ampleur dans le Web 2.0, surtout avec l'émergence des technologies REST et AJAX. En effet, dans Web 2.0, AJAX propose l'installation d'une application cliente AJAX qui permet de connecter directement le navigateur à un service distant, situé dans le serveur Web. Par conséquent, l'accès au service suit clairement une architecture client/serveur, avec un client plus ou moins lourd. De plus, selon la technologie REST, les services distants accédés directement par le client, sont des services de base CRUD à faible granularité permettant au client de faire des traitements basiques sur un ensemble de ressources. Par conséquent, l'approche Web est en train de se diriger de plus en plus vers une architecture, sûrement plus simple, mais de plus en plus client/serveur.

D'autre part, au niveau de la découverte de services Web, l'annuaire UDDI est conçu essentiellement pour mettre en œuvre un annuaire de services à l'échelle d'Internet. Cependant, l'UDDI soufre d'une manque de définitions au niveau des mécanismes de vérification et de maintenance continue, ce qui mène dans certains cas à découvrir un service qui n'est plus disponible ou qui est dégradé. Par conséquent, une inférence en temps réel est nécessaire pour mettre à jour la qualité des services ainsi que d'autres informations primordiales pour le bon fonctionnement de découverte.

En outre, l'approche Web est également confrontée aux défis de la composition de services. En effet, une des approches de composition est l'orchestration. Elle se base sur un orchestrateur central qui gère l'interaction entre les services afin de créer l'application finale. Cette approche centralisée renforce ainsi le problème d'étranglement et du point de défaillance au sein de la plate-forme de chaque fournisseur. D'autre part, si nous discutons la technologie BPEL qui est la plus déployée au niveau de l'orchestration de services, nous constatons qu'elle assure une faible granularité, ce qui est insuffisant dans le cas des processus complexes. De plus, même l'activité *Scopes* que propose BPEL pour regrouper des séquences d'invocation de services, facilite tout simplement l'écriture des processus, et elle ne permet pas à gérer en même temps des processus et des sous-processus.

Ayant présenté les défis de l'orchestration de services, nous concluons que l'approche de chorégraphie de services est plus adaptée à notre vision décentralisée pour la composition de services. Cependant, l'approche chorégraphique évoquée dans l'architecture Web néglige toute prise en compte du contexte de l'utilisateur, de la personnalisation de sa session de services, ainsi que la gestion de la QoS des services participants à la chorégraphie. En effet, la QoS des services n'est rajoutée à la description du contrat d'un service Web que dans la dernière version du WSDL (WSDL2.0). Pourtant, il n'y a pas une précision des mécanismes et des services qui vont gérer cette QoS. Ce même problème se pose aussi au niveau de la gestion de la mobilité des services Web, qui reste toujours un sujet non-traité.

3.3 Approche IT : SOA

Suite à l'évolution qu'a subit le monde de l'IT et l'augmentation des besoins qui a accompagné cette croissance, les sociétés doivent de plus en plus aligner leurs structures IT à ces nouveaux besoins, de façon à augmenter les revenus et optimiser les coûts. La création et l'intégration de processus métiers optimaux et opérationnels représentent ainsi l'objectif primordial de cet alignement. Ce défi d'intégration nécessite une technologie qui peut combler avec succès les besoins métiers et les besoins IT en se basant sur une solution viable qui ne doit pas seulement prendre en compte l'usage efficace et flexible de l'infrastructure IT, mais également être assez flexible pour s'adapter aux changements continus au niveau du processus métier et du modèle économique de chaque société. Le concept de l'orientation service semble être la meilleure solution pour répondre aux besoins métiers dans le cadre des systèmes distribués. Généralement, cette approche orientée service est efficace pour structurer des grands problèmes. En effet, le domaine IT considère que la logique requise pour résoudre des grands problèmes, comme les besoins des processus métiers, peut être mieux construite, exécutée, et contrôlée si elle est décomposée en des petites unités facilement gérables. Cette approche de décomposition en services n'aide pas seulement à la résolution des problèmes majeurs, mais elle facilite aussi l'ouverture des sociétés vers d'autres frontières du marché, et l'adaptation aux différents types d'applications qui se basent sur différentes technologies, plates-formes et modèles de données.

Afin de concrétiser cette approche de décomposition de la logique métier en services, le monde IT a conçu l'architecture orientée service, notée SOA (*Service Oriented Architecture*). Dans cette partie, nous commençons par décrire l'approche SOA (voir Sous-Section 3.3.1). Ensuite, nous discutons les principaux défis de cette approche SOA (voir Sous-Section 3.3.2).

3.3.1 Description de l'approche SOA

D'un point de vue métier, SOA est représentée par un ensemble de services et de processus flexibles conçus par une société afin de les exposer à leurs clients, partenaires ou même en interne pour les autres entités de cette société. Dans ce contexte, les mêmes services peuvent être recombinés et complétés par d'autres services afin de supporter les changements et l'évolution des besoins métiers au cours du temps. D'un point de vue technique, SOA est implémentée sous forme de composants de services qui sont invoqués pour effectuer un ensemble d'opérations spécifiques comme réponse à des tâches métiers spécifiques. Elle a ajouté la notion de contrat à la notion existante de fonction qui était considérée comme un bout de code spécifique qui effectue une tâche particulière. Cela rend la fonction indépendante de la technologie, mais spécifique au métier. Donc, en combinant ces deux points de vision, nous pouvons définir SOA comme un ensemble de fonctionnalités regroupées sous forme de services qui sont indépendamment décrits, et qui permettent ainsi de composer, d'une manière flexible, des processus métiers distribués et dynamiques, tout en étant réutilisés dans différents processus.

Au niveau des standards, il y a plusieurs organisations qui se sont impliquées dans la

standardisation d'un modèle de référence pour SOA. Nous citons par exemple W3C et OASIS dont chacune a sa propre culture et philosophie pour implémenter ses standards. Cependant, toutes les standardisations ont été influencées par les approches du marché commercial. Nous trouvons ainsi une grande contribution de la part des fournisseurs commerciaux (ex. *Microsoft*, *IBM*, *Oracle*, etc.) dans le processus de développement des standards SOA. Par conséquent, il n'y a pas qu'une seule organisation ou entreprise qui fournit et contrôle SOA. Cette dernière a évolué d'un ensemble de plates-formes propriétaires vers une architecture qui supporte des standards libres et des protocoles indépendants des fournisseurs. Ainsi, SOA garde son importance au sein du marché tant qu'elle conserve ses avantages auprès des différentes entreprises et qu'elle continue à avoir leurs supports. Nous listons ci-dessous les avantages garantis par SOA. Cette liste est non exhaustive et ne cite pas les améliorations techniques ; elle se contente des avantages métiers :

- Coexistence avec les structures existantes : SOA est une couche d'abstraction qui prend en compte les structures existantes dans le monde IT, et les représente sous forme d'un ensemble de services fournissant un ensemble de fonctions métiers. Par conséquent, les entreprises peuvent continuer à bénéficier des valeurs ajoutées de leurs ressources existantes, au lieu de recommencer à zéro leur structuration. Ainsi, le coût et l'effort pour intégrer les solutions antérieures et contemporaines sont diminués.
- Transparence : SOA n'est pas limitée à un ensemble spécifique de technologies comme c'était le cas des architectures existantes. Elle encourage l'application de plusieurs standards publics, ce qui améliore sa transparence.
- Alignement entre l'IT et les métiers : SOA met en exergue, auprès des cadres de niveau métier, l'intérêt et la valeur ajoutée du travail effectué par l'IT. Par conséquent, cette association de plus en plus proche entre le monde métier et l'IT pourra aisément justifier des nouveaux investissements au niveau IT.
- Flexibilité et faible TTM : SOA introduit une approche de services à couplages lâches, ce qui permet aux entreprises d'utiliser facilement ces services et de modifier rapidement leur logique de composition afin de s'adapter à tout changement au niveau des besoins métiers. Par conséquent, SOA est une approche flexible et efficace qui favorise une création rapide de nouveaux services métiers, et par la suite une diminution du TTM.
- Un développement à faible effort et faible coût : SOA fournit des services réutilisables qui remplissent certaines fonctions afin de répondre à certains besoins métiers, et qui sont susceptibles d'être réutilisés après pour répondre à d'autres demandes. Ainsi, cette approche diminue les efforts de développement et par conséquent les coûts de l'entreprise.

Ayant décrit et définit les avantages métiers apportés par SOA au sein du marché, nous détaillons dans la suite, ses aspects techniques ainsi que ses apports technologiques. En effet, SOA est une solution métier qui se base sur une entité technique nommée *Service*. Selon *Thomas Erl* [16], un *leader* du monde SOA, un service correspond à un ensemble d'opérations qui sont groupées logiquement et qui sont capables d'effectuer des unités reliées de travail. Afin de compléter leurs travails, les opérations

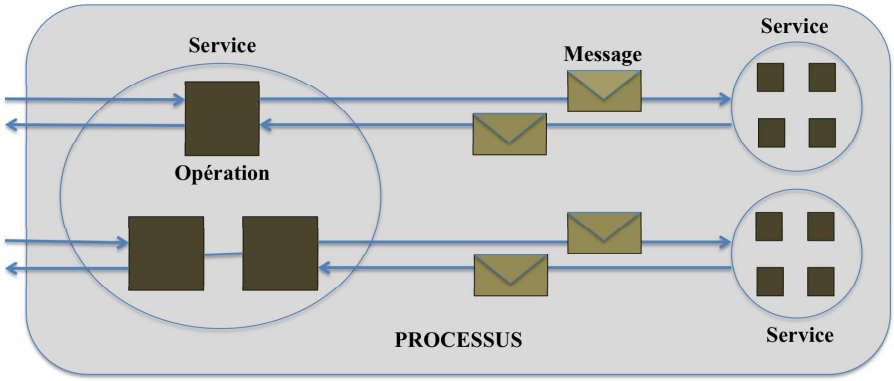

Fig. 3.6: Modélisation d'un processus SOA.

sont capables d'échanger des messages. Ces derniers représentent les données nécessaires pour compléter une partie ou bien la totalité d'une unité de travail. La dernière composante du modèle SOA est le processus. Il contient les règles métiers qui définissent les opérations qu'un service doit utiliser afin de compléter une unité d'automation. Il représente ainsi la coordination d'un ensemble de petites unités de travail afin de compléter une partie plus grande de travail. La Figure 3.6 représente le processus sous forme de services qui interagissent à l'aide de leurs opérations qui à leur tour échangent des messages. D'après ce modèle, nous constatons que les opérations d'un service peuvent être invoquées séparément et elles sont dépendantes de l'information reçue à l'entrée. Ainsi, pour chaque entrée au niveau du service, un traitement différent pourra être effectué. Par conséquent, un processus n'est pas nécessairement défini par ses services, puisqu'il peut se construire à partir d'une série d'opérations offertes par ses services. Autrement dit, chaque processus est défini partiellement par les opérations des services qu'il utilise. Cette modélisation montre bien qu'au niveau technique, nous sommes dans une architecture orientée opérations plutôt qu'orientée services.

Après avoir décrit les composantes d'une approche SOA, nous présentons l'ensemble des principes à respecter et à suivre dans toute conception et déploiement d'une architecture SOA [16] :

- Réutilisation : elle consiste à réutiliser un même service dans différents processus métiers, ce qui permet de diminuer les efforts de développement nécessaires pour répondre à des nouveaux besoins métiers. Comme nous l'avons déjà montré, un service est composé d'un ensemble d'opérations. Par conséquent, il faut que la logique encapsulée dans chacune de ces opérations soit réutilisable afin qu'on considère le service comme réutilisable en soi-même. Ainsi, plus les opérations d'un service sont génériques, plus le service sera réutilisé.
- Couplage lâche : elle consiste à limiter les dépendances dans les relations entre les services. Elle assure ainsi une agilité qui permet à une solution SOA de s'adapter d'une manière efficace à tout changement dans le monde IT. Normalement, ces changements sont dûs à des besoins externes à l'environnement IT et par conséquent sont difficiles à être planifiés à l'avance. Pour cette raison, avoir des relations à couplage lâche devient une propriété primordiale pour SOA. Elle est

assurée à l'aide des contrats de services qui permettent aux services d'interagir selon des paramètres prédéfinis, et de partager certaines connaissances tout en restant indépendants les uns des autres.

- *Stateless* : elle représente un service qui ne conserve pas les informations d'état et ne les gère pas. Si un service conserve un état pour une longue durée, il perdra ainsi sa propriété de couplage lâche, sa disponibilité pour d'autres requêtes, ainsi que son potentiel de passage à l'échelle. Pour cela, les services doivent être conçus d'une manière *Stateless* même si cela nécessite de déléguer la gestion des états à quelqu'un d'autre. Pour qu'un service soit *Stateless*, il faut que ses opérations soient conçues pour faire des traitements *Stateless*, c.à.d. le traitement d'une opération ne doit pas se baser sur des informations reçues lors d'une invocation précédente. Il faut noter que plus les messages échangés sont intelligents, plus les services seront *Stateless*.
- Autonomie : elle exige que la logique exposée par un service soit explicitement limitée. Cela permet au service d'effectuer l'auto-gouvernance de ses propres traitements sans être dépendant d'autres services. L'autonomie des services doit être prise en considération lors de la répartition de la logique métier en plusieurs services et lors du choix des opérations à regrouper pour constituer un service. Elle permet de remplacer un service dégradé par un autre sans perturber le fonctionnement du processus métier. Il faut noter que l'autonomie au niveau service consiste à séparer les frontières entre les services, mais elle n'empêche pas le partage des ressources sous-jacentes.
- Interopérabilité : elle constitue le bénéfice primaire des standards SOA. Elle est fondamentale pour toutes les autres propriétés. Elle permet aux services de communiquer et de collaborer entre eux afin d'offrir un service global, tout en gardant leurs autonomies.
- Abstraction : elle s'applique au niveau de l'interface du service qui apparaît ainsi comme une boxe noire qui cache à l'intérieur la logique interne et les détails qui ne sont pas décrits dans le contrat du service. Toutes ces informations sont invisibles par rapport aux demandeurs. Ainsi, la seule partie qui reste visible à l'extérieur du service est celle décrite dans son contrat.
- Description dans un contrat : elle correspond à une définition formelle représentée dans un contrat de service. Ce dernier facilite l'interaction entre les services en définissant : l'interface du service, ses caractéristiques, chaque opération, chaque message d'entrée/sortie supporté par chacune des opérations. Par conséquent, le contrat définie toutes les composantes SOA d'un service donné. En outre, le contrat peut aussi contenir des informations plus sémantiques qui expliquent comment un service va accomplir un certain travail.
- Découverte : elle aide à éviter une création accidentelle de services redondants ou même des opérations redondantes. Pour ce faire, elle se base sur un contrat de service, qui doit clairement décrire non seulement la logique générale du service, mais aussi la fonctionnalité offerte par chacune de ces opérations.
- Composition : elle consiste à composer des services en combinant d'autres services plus basiques. Cela permet de représenter la logique sur différents niveaux

de granularité et par suite faciliter la réutilisation et la création des couches d'abstraction. Ce principe attribué aux services garantit la conception de ces services de façon à les rendre susceptibles de participer efficacement dans d'autres composition de services, même si eux-mêmes sont composés par d'autres services afin d'accomplir leurs logiques. Cette propriété est étendue aussi au niveau des opérations qui doivent être conçues d'une manière standardisée et avec un certain niveau de granularité qui maximise les opportunités de composition.

D'après ce que nous venons d'évoquer, une architecture n'est SOA que si elle met en place des services qui respectent les principes cités ci-dessus. Cependant, les questions qui se posent ainsi sont : existe-t-il plusieurs catégories de services ? Comment les modéliser de façon à obtenir une architecture qui respecte bien les principes déjà évoqués ? Pour répondre à ces questions, nous nous intéressons à la typologie et à la granularité des services dans une architecture SOA. En effet, le niveau de granularité des services est un des facteurs essentiels du succès de SOA. Il a été introduit au début dans le monde des objets où il y avait les objets "à gros grain" (*coarse grain*) qui peuvent contenir d'autres objets, et les objets "à grain fin" (*fine grain*) qui contiennent quelques attributs. Ensuite, ce principe a été étendu au monde des services. Nous avons ainsi des services "à gros grain" qui accèdent à des services "à grain fin" pour effectuer leurs traitements. Cette granularité de services est ainsi très importante pour avoir une architecture SOA flexible et efficace, cependant, elle est considérée comme un principe très subjectif qui diffère d'une architecte SOA à une autre. Dans la suite, nous présentons deux différentes approches de modélisation des services SOA sur différents niveaux de granularité.

Une des premières approches de modélisation est celle de *Thomas Erl* [16]. Ce dernier s'appuie sur le principe d'abstraction afin de modéliser les services. Il considère

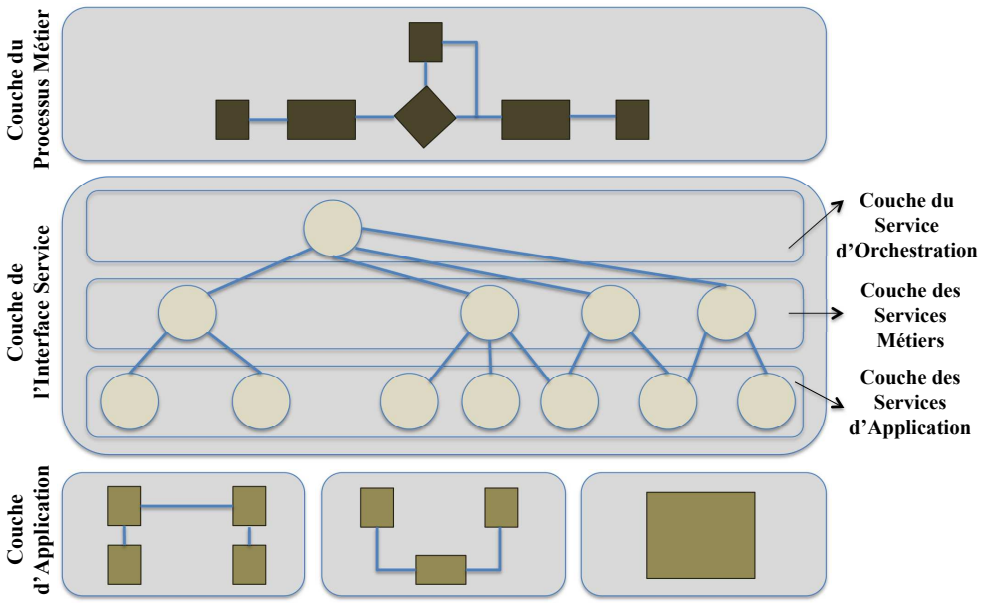

Fig. 3.7: Architecture SOA : première approche de granularité de services.

deux couches essentielles : la couche des applications représentant les logiques d'applications qui sont spécifiques à des contextes et à des types de plates-formes ; et au dessus, il y a la couche des processus métiers représentant les logiques métiers et assurant le plus haut niveau d'abstraction vis-à-vis des consommateurs finaux. Afin d'avoir une meilleure granularité, *Thomas Erl* a ensuite proposé une couche intermédiaire de services qui est elle même décomposée en trois sous-couches, comme le montre la Figure 3.7 : la sous-couche des services d'applications permettant l'accès aux applications sous-jacentes ; ensuite, il y a la sous-couche des services métiers explicitant sous formes de services la logique métier introduite par la couche des processus métiers ; et enfin, il y a la sous-couche du service d'orchestration permettant l'orchestration des services métiers sous-jacents afin de constituer la logique globale demandée.

D'autre part, une deuxième typologie de services est introduite par *Xavier Fournier-Morel et al.* [27]. Dans le but de différencier la vue métier de la vue technique, cette approche divise les services en deux grandes parties : les services métiers et les services techniques. Les services métiers sont conçus afin d'implémenter le modèle métier de services d'une entreprise et de représenter ainsi les différentes logiques métiers que les consommateurs finaux sont susceptibles d'invoquer. Pour effectuer sa logique, un service métier s'appuie sur plusieurs services techniques. Ces derniers donnent accès à des ressources techniques existantes (ex. la messagerie, l'imprimante, les bases de données relationnelles, etc.) tout en restant indépendant de ces ressources. La problématique qui se pose ainsi est celle du choix du meilleur niveau de granularité afin de modéliser le mieux possibles les services métiers définis. Pour ce faire, cette approche propose une typologie basée sur trois types de services, comme le montre la Figure 3.8. Le premier type correspond aux services CRUD qui associent chacun d'eux à un objet métier afin de créer, rechercher, mettre à jour et supprimer des informations concernant une ou

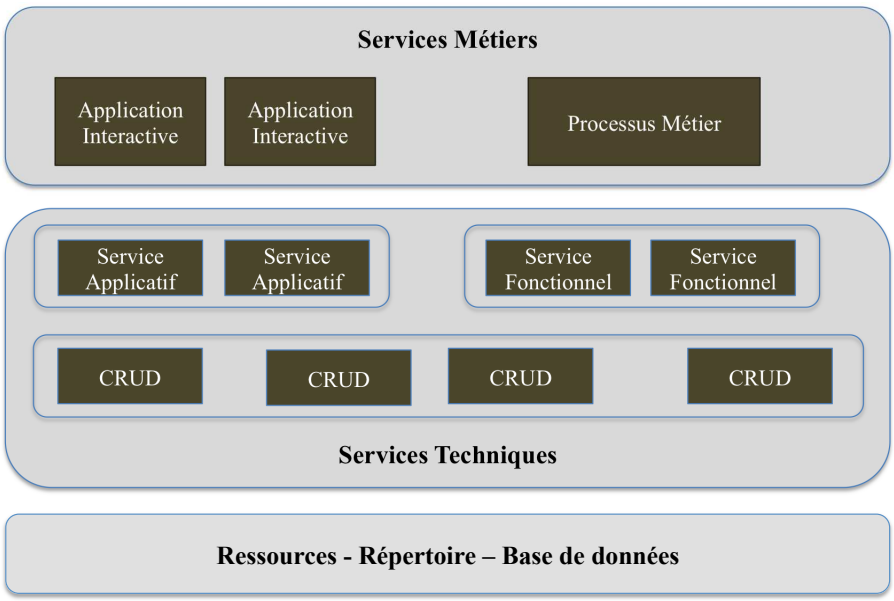

Fig. 3.8: Architecture SOA : deuxième approche de granularité de services.

plusieurs instances de ce même objet métier. Afin de masquer les services CRUD des applications interactives (ex. le portail) et afin d'alléger ces dernières, une couche de services applicatifs vient s'interfacer au-dessus de la couche de services CRUD. Par conséquent, cette couche introduit des services applicatifs réutilisables de haut niveau permettant aux concepteurs des applications interactives de ne pas se préoccuper des services CRUD "à grain fin". De même, afin de masquer les services CRUD du gestionnaire des processus métiers, un troisième type de services est conçu. Ces nouveaux services sont considérés de même niveau que les services applicatifs et ils sont nommés services fonctionnels. D'après leur nom, ils représentent le processus métier sous forme de fonctions métiers. Ils se présentent ainsi comme des services réutilisables de haut niveau permettant aux concepteurs des processus métiers de ne pas se préoccuper des services CRUD "à grain fin".

Comme nous l'avons déjà noté, il y a plusieurs standardisations pour SOA. Ainsi, cette dernière est modélisée en services selon différentes approches. Nous venons de décrire deux de ces approches mais il y en a plein d'autres. En effet, chaque groupe d'entreprises ou chercheurs aura sa propre vision de granularité et de décomposition des services. Cependant, toutes ces architectures sont obligatoirement guidées soit par une approche *top-down* soit par une approche *bottom-up*. L'approche *top-down* consiste à commencer par spécifier les besoins métiers qui se basent en grande partie sur les besoins des consommateurs. Ensuite, elle consiste à les décomposer en services métiers qui à leurs tours sont répartis en services de niveau plus bas, jusqu'à atteindre les services techniques de base ayant le plus faible niveau de granularité. D'après cette approche, lors de la modélisation d'un nouveau processus métier, un architecte SOA peut s'appuyer sur des services et même sur des opérations existantes et réutilisables. S'il n'en trouve pas, il procède à la création des services qui répondent à ces nouveaux besoins métiers. Selon l'approche *bottom-up*, ce sont les ressources internes que possèdent l'entreprise qui guident l'architecte. Ce dernier s'appuie ainsi sur des services bien connus et maîtrisés afin de créer des services de plus haut niveau. Cette approche est plus rapide que l'approche *top-down* dans la production, mais elle a une vision plus limitée au niveau de la conception, surtout lorsqu'il s'agit de la prise en considération des besoins des consommateurs.

Suite à ces deux approches, un nouveau défi surgit et se manifeste par un besoin de communication et d'interopérabilité entre les différents services conçus. Pour surmonter ce problème, un nouveau composant architectural nommé ESB (*Enterprise Service Bus*) est utilisé. Au début, l'ESB a été proposé par *IBM* en 2002, puis il a été adopté rapidement par la majorité des entreprises. De nos jours, il constitue une partie primordiale des architectures SOA existantes. L'ESB joue le rôle d'intermédiaire dans une interaction de services. Il correspond à une couche intelligente qui garantit une connexion à couplage lâche entre les systèmes d'information et les services distribués dans un environnement IT d'une entreprise. Il constitue un environnement d'intégration fortement distribué qui applique ses différentes capacités réparties sur plusieurs domaines comme la connexion, la transformation des messages, le routage fiable de ces messages, la médiation des transactions, l'orchestration du processus, et la sécurité. Ces caractéristiques sont essentielles pour garantir une intégration efficace des services

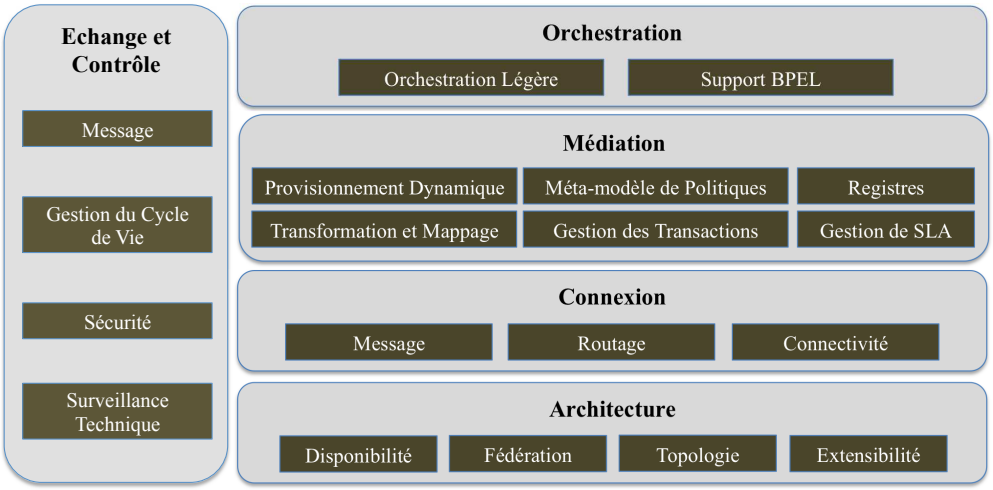

Fig. 3.9: Modèle architectural de référence d'un ESB.

dans une architecture SOA. Comme pour SOA, il y a différentes définitions et interprétations pour le fonctionnement d'un ESB. Afin de clarifier cette ambiguïté, l'équipe de recherche et de conseil IT chez la société internationale *Forrester* a défini [66], comme le montre la Figure 3.9, un modèle architectural de référence qui regroupe les différentes fonctionnalités d'un ESB en cinq parties fonctionnelles :

- Architecture : cette partie contient des fonctionnalités comme la topologie, la fédération, l'extensibilité et la disponibilité. La topologie d'un ESB consiste à déployer d'une manière distribuée des composants d'intégration à couplage lâche qui sont considérés comme des services accessibles par n'importe quel service connecté sur le bus. L'extensibilité est introduite par l'ESB au niveau des messages circulant afin de les rendre plus riches. La fédération est effectuée entre différents ESBs afin de permettre à un ESB de fonctionner avec d'autres environnements d'intégration qui sont gérés d'une manière plus répartie, tout en conservant son autonomie locale.
- Connexion : cette partie contient des fonctionnalités comme le support de plusieurs types de messages échangés, l'adaptation à différents types de protocoles de routage, ainsi que l'adaptation en suivant différentes possibilités de connexion.
- Médiation : cette partie assure le provisionnement dynamique des ressources, la description sous forme de meta-données des différents services, la découverte basée sur des registres, les politiques de gestion appliquées, la gestion évènementielle des transactions, ainsi que le respect du contrat de QoS signé avec l'utilisateur. De plus, l'ESB dans cette partie de médiation permet de transformer les formats des messages échangés ainsi que d'agréger et de décomposer les sémantiques de services afin que chaque participant à une conversation soit inconscient des détails de traitement des autres.
- Orchestration : cette partie permet à l'ESB d'orchestrer des services connectés au bus, soit d'une manière légère, soit d'une manière robuste à base de BPEL. Ces services peuvent être locaux ou même distants.

– Changement et contrôle : cette partie assure des outils de conception de services et des fonctionnalités de sécurité comme l'identification, l'authentification, l'autorisation d'accès ainsi que d'autres politiques de sécurité plus avancées. De plus, cette partie comporte des fonctionnalités de gestion qui tracent le cycle de vie d'un service, du développement, déploiement, intégration, réutilisation, gestion et optimisation. Enfin, malgré l'objectif de rendre l'ESB autogérable, certains problèmes peuvent nécessiter une intervention humaine pour les régler, pour cette raison, cette partie introduit des fonctionnalités de contrôle et de gestion.

Ces différentes fonctionnalités assurées par l'ESB rendent ce dernier, un composant primordial pour répondre aux différents besoins d'intégration et pour supporter les efforts d'implémentation des architectures SOA. Cependant, la mise en place d'un ESB ne fait pas émerger spontanément les services et processus métiers. L'émergence de ces derniers consiste à identifier des services à valeurs ajoutées, de vérifier s'ils répondent aux besoins qui émergent, et réciproquement que les besoins émergeant ciblent bien les services existants. Au cours de cette émergence des services à base SOA, plusieurs fournisseurs l'ont confondus avec les services Web. Cet amalgame est dû au rapprochement entre ces deux approches au niveau de leurs piles technologiques. Comme c'est le cas de l'architecture des services Web, l'approche SOA est représentée par un ensemble de couches complémentaires que nous décrivons, de bas en haut, comme suit : la couche de transport, qui contient des mécanismes d'échanges des messages entre le fournisseur et le consommateur ; la couche des messages, qui établit le format des messages échangés ; la couche de description, qui décrit les interfaces d'un service, sa nature, ses opérations et ses entrées et sorties ; la couche de découverte, qui s'appuie sur un registre dans lequel sont enregistrés les descriptifs des services ; la couche du processus métier, qui s'appuie sur une abstraction de sous-couches de services et qui permet la composition et la réutilisation des services basiques pour former des services métiers plus complexes ; et à la fin, il y a des couches verticales qui assurent un support commun de gestion de la QoS, de sécurité, de gestion par politiques, etc. pour toutes les couches précédentes.

Malgré cette ressemblance au niveau des piles technologiques, la différence est claire entre ces deux approches. En effet, à l'intérieur du monde IT, une approche SOA peut très bien se réaliser sans le recours aux services Web. De même, par une réflexion à l'inverse, le déploiement des services Web ne veut pas dire directement que tous les principes SOA sont bien respectés. D'une part, les services Web ne sont pas automatiquement réutilisables, autonomes, *stateless* et capable d'être découverts. Un effort de modélisation et de conception doit être réalisé afin d'assurer ces principes au niveau des services Web. D'autre part, l'approche REST ne se base pas sur une typologie de services pertinente, et par conséquent, elle ne respecte pas le principe de granularité des services adoptée par SOA. Pour conclure, nous pouvons considérer que les services Web correspondent à un ensemble spécifique de standards et de spécifications qui aident à la mise en œuvre d'une architecture SOA. Les services Web sont considérés comme complémentaires à SOA, pour les raisons suivantes : premièrement, ils fournissent un standard libre et un modèle compréhensible par les machines, aidant ainsi à la description des interfaces de services d'une manière explicite et indépendante des implémentations ; deuxièmement, ils fournissent des mécanismes de communication

qui sont transparents vis-à-vis de la localisation et qui favorisent l'interopérabilité ; et finalement, ils supportent une implémentation des services réutilisables avec une très grande flexibilité, en se basant sur des technologies comme BPEL, WSDL, SOAP, etc.

De nos jours, l'approche SOA n'est plus limitée au domaine IT. Elle est aussi largement adoptée dans le monde des télécommunications et dans le monde de l'Internet et du Web, puisqu'elle permet de réduire le délai d'accès au marché en favorisant une composition lâche, rapide et flexible des nouveaux processus transverses à partir d'une typologie de services distribués. De plus, l'usage de SOA dans l'industrie est de plus en plus favorisé, grâce à son approche avantageuse de valorisation et de réutilisation des systèmes existants. En effet, afin de permettre l'accès des services métiers SOA aux informations et traitements implémentés par les systèmes existants, l'approche SOA introduit des services métiers à usage particulier, nommés services façades [27]. Ces derniers permettent l'accès des services clients de haut niveau aux opérations existantes sans se soucier de l'hétérogénéité technologique. Il y a deux types d'architectures d'intégration des services façades dans les systèmes existants : une est décentralisée et l'autre est centralisée. L'intégration décentralisée consiste à créer des services façades (ex. sous forme de services Web) au sein de chaque système existant. Dans cette approche, les services façades sont considérés comme des services légataires locaux qui seront invoqués d'une manière homogène par les services de haut niveau SOA afin de pouvoir communiquer avec les services existants. L'intégration centralisée consiste à créer des services façades au sein du système central, c.à.d. au sein de l'architecture SOA. Ces services sont considérés comme des services techniques de bas niveau qui permettent l'accès direct au support de communication avec les systèmes existants distants.

3.3.2 Défis de l'approche SOA existante

Au contraire des autres approches, le domaine IT était conscient de la nécessité d'une solution architecturale bien structurée qui intègre les avancements existants et qui soit ouverte pour toute sorte d'évolution dans le monde des télécommunications. Pour cette raison, il a conçu l'architecture SOA. Cependant, de nos jours, les architectures SOA existantes connaissent encore quelques défis à lever.

En effet, les architectures SOA de nos jours ne traitent pas le défi de convergence de services d'un point de vue conceptuel, mais plutôt selon une approche adaptative à l'aide des services façades implémentées. Selon notre réflexion, nous trouvons insuffisant le fait de contourner le défi de convergence de services en introduisant des services intermédiaires qui assurent une transformation des formats de messages. Si nous faisons une simple analogie avec la convergence qui est faite au niveau réseau, nous constatons que cette dernière ne se limite pas à la mise en place des passerelles entre les différents réseaux d'accès d'une part et le réseau cœur d'autre part pour permettre un passage unique tout-IP aux différents services. La convergence réseau est plutôt une approche conceptuelle que les fournisseurs de services adoptent de façon à exposer leurs services indépendamment du média que va utiliser le consommateur pour y accéder. Suite à cette analogie, nous proposons par la suite une approche de réingénierie de services,

qui traite la convergence de services à un niveau conceptuel plus élevé que celui d'une simple adaptation de format.

En outre, SOA se confronte aussi à des défis au sein de son modèle de service. Ce dernier, comme nous l'avons déjà discuté dans cette partie, est basé sur un ensemble d'opérations dont chacune est considérée comme une unité de travail qui effectue une fonction spécifique et qui peut être invoquée séparément des autres opérations d'un même service. D'après la Figure 3.6, nous constatons que les opérations à exécuter dépendent de l'information reçue à l'entrée du service. Ainsi, un même service peut avoir différents traitements selon les différentes combinaisons d'opérations exécutées. Par conséquent, nous pouvons conclure qu'au niveau technique, l'architecture SOA existante est une architecture orientée opérations plutôt qu'une architecture orientée services. Dans notre proposition, nous allons concevoir un nouveau modèle où l'approche service sera plus respectée.

En plus de son modèle de service, l'approche SOA existante connaît certains défis au niveau de son ESB. D'après une enquête menée par *Ken Vollmer et al.* [66] au près de 74 architectes d'entreprises et gestionnaires de développement d'applications utilisant ESB, nous constatons que la majorité d'entre eux utilisent l'ESB pour ses fonctionnalités de routage (95%) et d'échanges de messages (92%). De plus, un grand pourcentage d'utilisateur l'utilise pour la transformation de données (77%) ou comme un médiateur (58%). D'après cette étude, nous pouvons conclure qu'au niveau technique d'usage, les ESB existants sont introduits dans les architectures SOA pour un simple usage de médiation et pour assurer l'interopérabilité entre les services. Nous proposons par la suite, une approche de bus plus évoluée au niveau conceptuel.

Le dernier défi à souligner au niveau des architectures SOA existantes est son manque de cohabitation avec l'approche NGN sous-jacente et d'adaptation aux besoins temps réel de ses utilisateurs. En effet, l'approche SOA de nos jours est orientée fournisseurs et non pas utilisateurs finaux. Elle répond aux besoins des entreprises au niveau de la structuration des services, sans s'intéresser aux préférences de ses utilisateurs ainsi que leurs contextes ambiants. De plus, SOA interprète l'approche service indépendamment de la vision session de bout-en-bout et par suite, sans tenir compte de certains défis comme l'hétérogénéité des réseaux sous-jacents et la mobilité.

3.4 Conclusion

Pour conclure, nous présentons dans la Figure 3.10 un tableau récapitulatif qui cite l'ensemble des avancements qu'a apportés chacune des architectures de services existantes, ainsi que leurs limitations majeures.

Grâce à ce tableau, nous constatons que les architectures de services ont évolué en première étape des approches monolithiques Telco vers une approche chorégraphique décentralisée introduite par les services Web. Cependant, ces derniers conservent une relation verticale client/serveur lors de l'accès aux services. Pour dépasser cette limitation, l'approche SOA a amélioré la chorégraphie en introduisant une approche distribuée horizontale qui supporte une composition de services à différents niveaux de granula-

rité. Elle introduit aussi des services réutilisables, interopérables, autonomes et à liens lâches. Par conséquent, nous pouvons conclure que SOA correspond le mieux à nos besoins de composition horizontale de services et propose la version la plus avancée au niveau du modèle de service. Cependant, comme le montre le tableau, plusieurs défis limitent l'usage de cette approche comme une architecture de services de nouvelle génération. Afin de lever ces défis, nous avons besoin d'un nouveau modèle architectural qui garantit, d'une part, la convergence des services (Telco, Web et SOA), et d'autre part, la personnalisation de la session de services composée en horizontal, tout en tenant compte des défis de mobilité et d'hétérogénéité qui relèvent du réseau sous-jacent. Ce nouveau modèle architectural sera le point focal du chapitre suivant.

		Valeurs Ajoutées	Limites
Telco	RI	- Processus d'appel intelligent: traitement substitutif à base de SIBs. - Notion de services: association des SIBs à des services applicatifs.	- Pas d'identification des services applicatifs. - Manque de personnalisation de services.
	OSA	- APIs ouvertes standardisées qui favorisent la portabilité des applications dans des réseaux hétérogènes.	- Approche monolithique, client/serveur basée sur des APIs. - SCS centralisé => étranglement.
	OMA	- *Enablers* réutilisables: éviter les problèmes de Silos.	- Réutilisation des *enablers* contrainte par des liens forts.
	IMS-SCIM	- Orchestration des éléments de services à partir de la couche de contrôle IMS et de la signalisation SIP.	- Approche monolithique, client/serveur. - SCIM centralisé => étranglement. - Manque de standardisation du SCIM et manque de spécifications du mécanisme d'interaction entre les serveurs SIP.
Web		- Amélioration de la sémantique de création et de collaboration des services, en introduisant des *Web Services APIs*. - Chorégraphie : vision décentralisée pour la composition de services. - Web 2.0 : amélioration de l'interaction de l'utilisateur avec le portail Web.	- Approche client/serveur basée sur des *Web Services APIs*. - Au niveau protocolaire: REST favorise l'approche client/serveur et l'accès à des services CRUD à faible granularité. - Orchestration centralisée. - Découverte d'après l'UDDI de services qui peuvent être dégradés ou indisponibles <=> Manque d'inférence dynamique.
IT (SOA)		- Structure ouverte et distribuée. - Urbanisation : services à différents niveaux de granularité => composition des processus métier complexes. - Services réutilisable, interopérable, autonome et à liens lâches. - ESB: médiation des services à différents niveaux de granularité.	- Usage des services de façades pour l'adaptation avec les services d'autres domaines => Mauvaise vision de la convergence. - Modèle de service: approche orientée opération. - ESB: limité à son rôle de médiateur. - Pas de personnalisation de services: Pas de prise en compte des préférences des utilisateurs, Pas de gestion de QoS et de mobilité.

Fig. 3.10: Tableau de synthèse des architectures de services existantes.

4 Proposition

Dans ce chapitre, nous levons les deux premiers verrous identifiés suite à l'émergence des contextes NGN/NGS et de l'approche *user-centric*. Pour ce faire, nous repensons le modèle de services, et nous proposons un nouveau modèle architectural (voir Section 4.1) représenté par une architecture de services, dénommée NGN/NGS Middleware (voir Section 4.2).

4.1 Modèle Architectural

Pour accompagner l'évolution des réseaux vers le NGN, l'évolution des services est impérative, d'où les travaux de conception d'une nouvelle génération de services, notée NGS. Au cours des années, plusieurs modèles architecturaux ont été proposés pour accompagner cette évolution des besoins. Cependant, comme nous venons de le démontrer, les architectures de services existantes restent obsolètes vis-à-vis des exigences à venir. En effet, les architectures Telco conservent toujours leurs approches monolithiques, verticales et fortement liées aux technologies du réseau sous-jacent. L'approche Web a pu surmonter le problème des architectures en silos, mais elle connaît toujours des défis au niveau protocolaire, surtout avec l'émergence du Web 2.0 qui soutient de plus en plus l'usage du protocole REST favorisant l'approche client/serveur pour accéder aux ressources, aux dépens d'une approche de granularité, d'abstraction et de composition de services. Enfin, l'approche SOA semble le candidat le plus adéquat pour structurer les services de l'avenir. Cependant, les architectures SOA existantes se confrontent à des défis au niveau de la conception des services et au niveau de la gestion des besoins et des préférences des utilisateurs.

Afin de lever ces verrous technologiques et conceptuels que nous avons identifiés, nous proposons un nouveau modèle architectural qui s'appuie sur un nouveau modèle de services. Notre nouveau modèle architectural permet la structuration des services tout en se basant sur trois approches technologiques unifiées :

- Une approche SOA améliorée (voir Sous-Section 4.1.1)
- Une approche EDA (*Event Driven Architecture*) (voir Sous-Section 4.1.2)
- Une approche horizontale sous forme de Middleware (voir Sous-Section 4.1.3)

4.1.1 Approche SOA améliorée

La notion de service constitue le premier pilier de notre modèle architectural. L'ISO 20000 (*International Organization for Standardization*) définit le service [39] comme "une prestation composable qui doit être source de valeur pour le consommateur et le fournisseur". D'après cette définition, toute architecture qui désire répondre d'une manière flexible et efficace aux besoins des utilisateurs et des fournisseurs doit obligatoirement supporter une composition dynamique de services. Selon l'approche *user-centric*, cette composition de services doit aussi être personnalisée et doit converger des services appartenant à différents domaines (Telco, Web et IT). Pour ce faire, nous proposons un nouveau modèle de service qui est applicable pour n'importe quel contexte technologique et dans n'importe quel contexte d'usage. Par conséquent, nous améliorons l'approche de service supportée par l'architecture SOA, en redéfinissant les services pour être conformes aux caractéristiques suivantes :

- Composition à couplage lâche : la composition consiste à constituer un service globale à partir d'un ensemble d'éléments de services composables. Cette composition représente la logique de services demandée par un utilisateur. Par conséquent, elle est personnalisée, contrairement aux éléments de services qui la constituent et qui sont conçus et déployés indépendamment des contextes et des préférences des utilisateurs. Afin de supporter la personnalisation de services, cette composition s'appuie sur des couplages lâches entre les éléments de services. Cela permet de remplacer un élément de service par un autre et d'adapter, d'une manière dynamique et transparente, la logique de services à tout évènement temps réel provenant de la part de l'utilisateur, du fournisseur ou bien de l'élément de service lui-même.
- Autonomie : elle consiste à concevoir, déployer et puis exécuter des éléments de services qui soient indépendants les uns des autres. Selon le modèle de service appliqué aujourd'hui dans les architectures SOA existantes, le service est considéré comme un ensemble d'opérations dont chacune est une unité de travail à valeur ajoutée qui effectue une fonction spécifique et qui peut être invoquée séparément des autres opérations d'un même service. Par conséquent, un même service peut avoir différents traitements selon les différentes combinaisons d'opérations exécutées. Contrairement à cette approche orientée opération, qui augmente les dépendances, nous favorisons plutôt dans notre modèle l'approche orientée service. Ainsi, chaque élément de service est considéré comme une unité de travail atomique à valeur ajoutée. Il est conçu comme une chaîne fixe d'opérations, et il est modélisé comme une boite noire qui n'a aucune contrainte ou condition pour son enchaînement et qui rend une fonction et une QoS bien déterminées. Cette autonomie favorise aussi le remplacement d'un élément de service par un autre sans perturber le fonctionnement des autres éléments de services participant à la logique de services personnalisée.
- Autogestion : elle consiste à avoir des éléments de services qui sont capables de contrôler et de surveiller leurs propres paramètres de QoS courante. Ainsi, chaque élément de service peut vérifier en temps réel si son comportement est conforme

ou non à celui négocié dans le contrat de QoS préétabli. Pour garantir cette autogestion, un agent de QoS est associé à chaque élément de service. Cet agent compare en temps réel sa QoS courante à une valeur seuil à ne pas dépasser, et selon le résultat, il envoie comme notification soit un *IN-Contract* soit un *OUT-Contract*.

- Réutilisation : elle consiste à concevoir des éléments de services qui soient de plus en plus génériques afin de favoriser leurs réutilisations dans différents processus, ce qui permet de diminuer les efforts de développement nécessaires pour répondre à des nouveaux besoins.
- Mutualisation : au-delà de la réutilisation d'un élément de service, c'est la partageabilité d'une ressource service dont il est question dans la NGS. Pour cela, nous proposons la conception d'un élément de service non seulement réutilisable mais aussi mutualisable. Ce dernier est ainsi considéré comme une ressource ayant une capacité bien déterminée, et qui est capable de partager simultanément cette capacité entre différents utilisateurs. Ceci permet d'optimiser l'utilisation de la ressource service. Cependant, le partage d'une ressource ne doit pas l'empêcher de respecter ses contrats de QoS avec chacun de ses utilisateurs simultanés. Pour cette raison, nous définissons une file d'attente au sein de chaque élément de service. Cette file représente en terme de QoS courante le potentiel de la ressource service à accepter ou refuser une nouvelle requête. En effet, même si l'élément de service est en train de traiter une requête, il peut stocker dans sa file d'attente d'autres requêtes à condition que ça ne dépasse pas sa capacité courante de stockage. Ainsi avec ce mécanisme nous pouvons partager dynamiquement le service entre plusieurs utilisateurs tout en garantissant la QoS demandée par chaque requête et en utilisant au maximum la QoS offerte par le service.
- *Stateless* : pour avoir des éléments de services mutualisables entre plusieurs utilisateurs, nous avons besoin de concevoir des éléments de services qui réalisent les mêmes opérations pour tous les utilisateurs sans aucune conservation de données ou d'état spécifiques à ces derniers. Pour atteindre ce but, nous concevons des éléments de services *Stateless* qui exécutent des traitements génériques pour tous les utilisateurs, indépendamment de leurs contextes et sans maintenir leurs données spécifiques. Cette caractéristique favorise ainsi la possibilité de remplacement dynamique d'un élément de service par un autre dans la session de services. Il faut noter que le fait d'avoir des éléments de services *Stateless* ne veut pas dire que la sémantique de l'utilisateur soit négligée, et que l'autonomie va être favorisée aux dépens de l'intelligence de la couche service. En effet, c'est dans la composition de services que le contexte et les préférences de l'utilisateurs sont pris en compte. Par conséquent, c'est la session de services qui va être *Statefull* et qui va être spécifique à chaque utilisateur. Cela implique que la sémantique de chaque utilisateur sera maintenue tout au long de sa session de services.
- Interfonctionnement : elle consiste à assurer non seulement l'interopérabilité mais aussi l'interconnexion entre ces éléments de services. En effet, l'interopérabilité consiste à avoir des éléments de services autonomes qui collaborent entre eux afin de composer un service global. Quand à l'interconnexion, elle représente la mise

en relation entre ces différents éléments de services en définissant des interfaces et des liens génériques. La prise en considération de ces deux propriétés maintient un interfonctionnement dynamique entre plusieurs éléments de services tout en évitant les conflits de traitement et de *deadlocks*.

4.1.2 Approche EDA

Le deuxième pilier de notre modèle architectural est l'approche évènementielle EDA [46]. En effet, l'approche SOA toute seule ne constitue pas une réponse suffisante aux problématiques d'adaptation dynamique et de maintenance transparente face à la grande instabilité du contexte et des préférences de l'utilisateur, et face aux changements temps réel au sein même de la plate-forme de services. Typiquement, le modèle d'interaction le plus commun dans les approches SOA actuelles est celui des interactions synchrones sous forme de requêtes/réponses. Ce modèle favorise l'usage de services interactifs dépendants qui sont incapables de prendre des décisions face à des évènements temps réel.

Pour dépasser ce défi, nous avons besoins de définir d'autres types d'interactions que ceux proposés par l'approche actuelle SOA, d'où l'intégration de l'approche EDA dans notre modèle architectural. Selon cette approche évènementielle, les services deviennent des vrais automates accessibles après n'importe quel évènement. En effet, lorsqu'un évènement aura lieu, seuls les services qui sont abonnés à ce type d'évènements seront notifiés, et par suite, ces services exécutent les actions correspondantes. L'action peut être une invocation d'un autre service, la mise à jour d'une information dans une base de connaissances décisionnelles, ou bien une création d'un nouvel évènement.

Par conséquent, nous améliorons notre modèle architectural en combinant notre approche SOA améliorée avec une approche EDA qui supporte les deux propriétés suivantes :

- Paradigme asynchrone de publication/souscription : cette approche asynchrone permet d'avoir une communication à sens unique entre la source qui publie l'évènement et le consommateur qui a souscrit à ce type d'évènement. Par conséquent, les notifications subissent un *Push* de la part de la source d'évènement, et non pas un *Pull* de la part du consommateur de cet évènement. Ainsi, seule la source est capable de déterminer à quel moment elle doit envoyer la notification. Elle ne s'intéresse pas à la disponibilité du récepteur, et elle n'attend pas une réponse de sa part, contrairement à l'approche synchrone requêtes/réponses.
- Découplage entre la source et la destination de l'évènement : cette caractéristique consiste à l'indépendance des sources et des récepteurs des évènements ; ils sont inconscients les uns des autres. La source est seulement consciente de l'évènement. Elle n'a aucune connaissance des identités des récepteurs qui ont souscrit à cet évènement et des traitements effectués par la suite sur cet évènement. Autrement dit, la source envoie une notification et non pas une requête, ainsi elle ne peut pas spécifier dans sa notification la logique ou bien l'action que le récepteur doit effectuer. Par conséquent, ce principe favorise de plus en plus la composition de services à couplages lâches qui est adoptée dans notre modèle architectural.

4.1.3 Approche horizontale sous forme de Middleware

Afin de structurer l'ensemble des concepts qui caractérisent notre approche évènementielle orientée service, nous proposons un modèle architectural qui consiste à étendre l'approche horizontale des couches réseaux vers la couche service et de supporter, en plus de la convergence réseau, une convergence au niveau des services. Par conséquent, cette approche horizontale convergente permet de rendre homogène et transparente la structure architecturale de bout-en-bout, vis-à-vis des utilisateurs et des fournisseurs. En outre, l'approche horizontale orientée service que nous adoptons permet de pallier les problèmes de flexibilité des architectures verticales en silos et les problèmes de couplage fort des approches distribuées orientées objet. Pour ce faire, notre modèle architectural favorise la distribution de services autonomes et multi-domaines, et permet leur composition d'une manière dynamique en une logique de services trans-organisationnelle et à couplages lâches.

L'une des complexités majeures introduites par la composition horizontale d'un service global à partir d'un ensemble d'éléments de services indépendants est, comment supporter et rendre transparent la distribution de ces éléments, surtout que nous sommes face à des systèmes distribués de plus en plus instables (mobiles). Ces derniers sont constitués d'un ensemble d'hôtes mobiles qui évoluent dans un contexte extrêmement dynamique, ce qui rend instable et à qualité variable l'interconnexion des éléments de services distribués. Afin de faciliter la tâche des concepteurs de services, il faut alors fournir une transparence vis-à-vis de la distribution et de la localisation, leur permettant d'accéder à l'ensemble des éléments de services locaux ou distants d'une manière générique, et par suite de dépasser la complexité introduite par la distribution des ser-

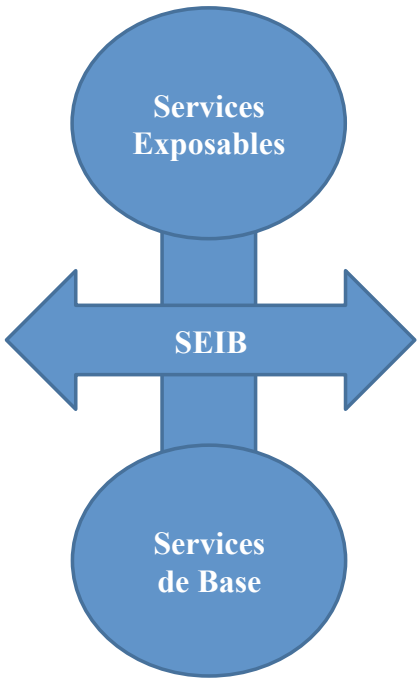

Fig. 4.1: Modèle de base du *Middleware*.

vices. Pour atteindre ce but, nous nous appuyons sur une approche de *Middleware*. Cette modélisation horizontale sous forme de *Middleware* favorise l'interaction dynamique entre des services distribués et facilite la composition transversale de la logique de services.

En effet, le modèle générique de notre *Middleware* (voir Figure 4.1) est structuré en trois parties dont chacune joue un rôle essentiel dans le déverrouillage des défis au niveau de la maîtrise du contexte *user-centric*, de la cohabitation avec les réseaux NGN, de la convergence et de la personnalisation introduite par NGS. Ces trois parties architecturales sont :

- Le SEIB (*Service Elements Interconnection Bus*) : c'est un bus d'interconnexion entre plusieurs éléments de services distribués. Il supporte d'une part, une composition de services horizontale, personnalisée, dynamique et à couplages lâches, et d'autre part, une agrégation verticale de tous les composants qui interviennent sur les quatre niveaux de visibilité d'une session (utilisateur, équipement, réseau, service).
- Les Services de Base : cette partie située au-dessous du SEIB contient un ensemble de services communs à tout le monde. Elle comporte des services IT (facturation, etc.), des services de sécurité, des services de gestion de QoS, de mobilité et d'hétérogénéité, ainsi que des services pour supporter la personnalisation de la logique de services. Par conséquent, cette partie intègre des services de bases qui rendent transparent le réseau sous-jacent. Ils favorisent aussi la convergence NGN/NGS, en supportant d'une part l'approche NGS, et en répondant d'autre part aux défis des contextes NGN et de l'approche *user-centric*.
- Les Services Exposables : cette partie située au-dessus du SEIB contient des services applicatifs à valeurs marchandes. Ils sont normalement créés suite à une composition de services incluant des services de base. Comme leur nom l'indique, ils seront exposés dans les catalogues des fournisseurs de services et des opérateurs pour que les utilisateurs puissent les solliciter. Il faut noter que n'importe quel utilisateur peut aussi exposer des services et ainsi jouer le rôle de fournisseur.

Après avoir décrit notre modèle architectural qui suit une approche SOA/EDA, horizontale et sous forme de *Middleware*, nous nous intéressons dans la suite de ce chapitre aux fonctionnalités et au processus qui doivent être introduits dans notre architecture de services afin de lever les principaux verrous identifiés. Par conséquent, nous présentons dans la suite, d'une manière plus explicite et plus détaillée, notre modèle architectural dénommé NGN/NGS Middleware.

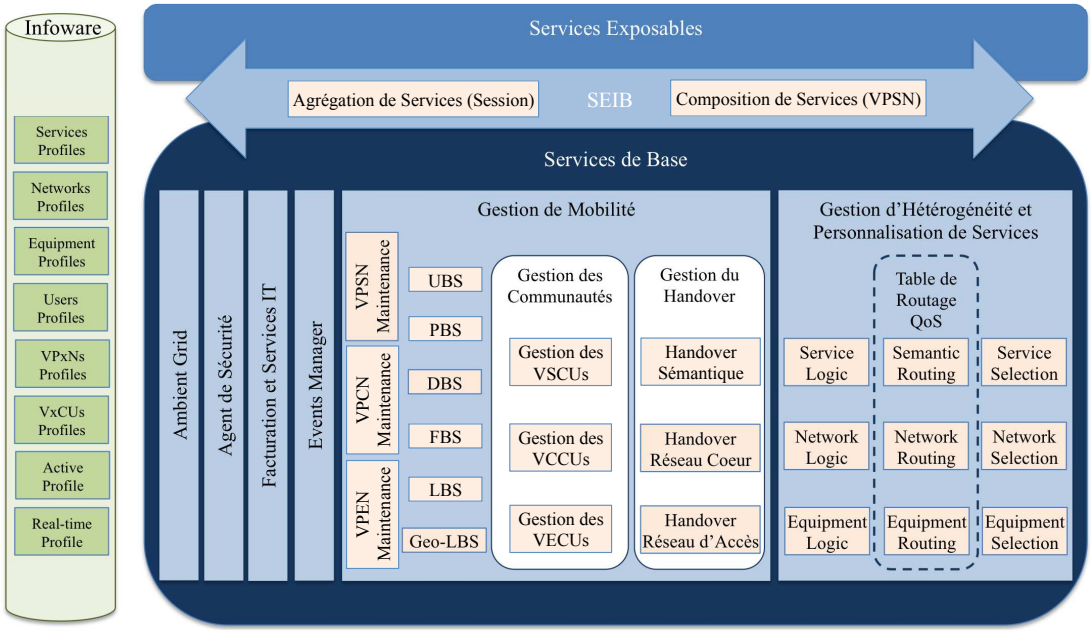

Fig. 4.2: Architecture du NGN/NGS Middleware.

4.2 NGN/NGS Middleware

Nous proposons dans cette partie un NGN/NGS Middleware (voir Figure 4.2) qui s'appuie sur une architecture SOA/EDA et qui se base sur un ensemble de composants fonctionnels développés sous forme de services. Ces derniers ont comme rôles de supporter le contexte *user-centric*, de dépasser les défis au niveau NGN (mobilité et hétérogénéité) et d'introduire une nouvelle vision de services NGS qui favorise la personnalisation et la convergence de services. Le but est d'obtenir, pour chaque utilisateur, une session unique, convergente, personnalisée, continue, dynamique, transparente et trans-organisationnelle. Comme son nom l'indique, le NGN/NGS Middleware proposé suit une modélisation sous forme de *Middleware* (Services de Base, SEIB, Services Exposables).

4.2.1 Les services exposables

La première partie de notre architecture est celle des services exposables. Ces derniers peuvent être soit des services applicatifs élémentaires qui ne nécessitent aucune composition, soit des services applicatifs composables. Ces derniers sont ainsi créés à l'aide d'un ensemble de services de base qui peuvent être composés avec d'autres services exposables. Puisque l'exposition de services est la dernière étape de l'intégration pour associer les services aux stratégies métiers des fournisseurs, les niveaux de granularité et les décisions d'exposition de services vont ainsi dépendre des modèles métiers adoptés par ces derniers. Comme nous l'avons déjà mentionné, les services exposables seront exposés dans les catalogues des fournisseurs de services et des opérateurs pour que les utilisateurs puissent les solliciter. De plus, tout utilisateur peut aussi jouer le rôle de fournisseur en exposant ses propres services.

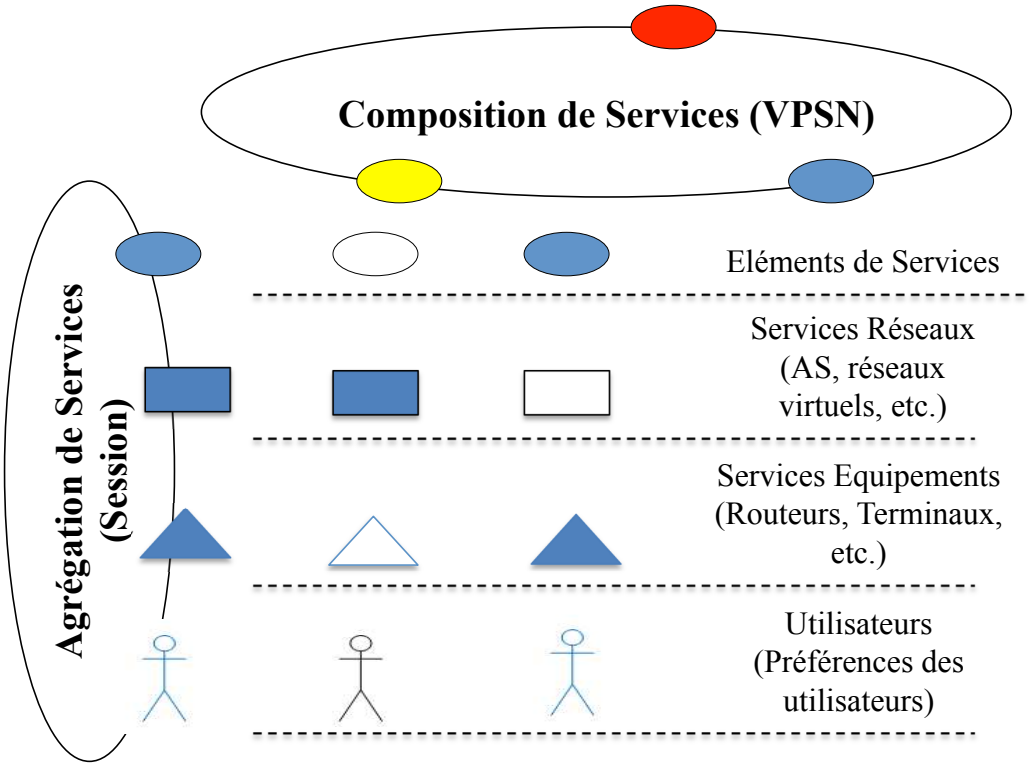

Fig. 4.3: SEIB : composition et agrégation de services.

4.2.2 Le bus d'interconnexion SEIB

La deuxième partie de notre architecture est celle du bus d'interconnexion SEIB. Comme son nom l'indique, le *Service Elements Interconnection Bus* a pour rôle d'interconnecter les éléments de services. C'est un ESB amélioré qui ajoute à sa fonction de médiation et d'interopérabilité entre les éléments de services une fonctionnalité d'interconnexion virtuelle. En effet, comme nous l'avons montré dans le Chapitre 3, au niveau technique d'usage, les ESB existants sont introduits dans les architectures SOA simplement pour leurs fonctionnalités de médiation et d'interopérabilité entre les services [66]. Cependant, fournir des éléments de services aux utilisateurs de façon transparente est parmi les besoins les plus importants que les fournisseurs de services doivent satisfaire dans le contexte NGN/NGS *user-centric*. Pour cela, tout architecture de services doit se doter, au niveau de son ESB, de mécanismes offrant à l'utilisateur la transparence vis-à-vis de la localisation des services distribués ainsi que de la maintenance de leur QoS et de leur continuité. Pour répondre à ce besoin de transparence, nous ajoutons, au sein de notre SEIB, la fonctionnalité d'interconnexion virtuelle entre les services. Comme le montre la Figure 4.3, l'interconnexion virtuelle se manifeste au niveau horizontal sous la forme d'une composition de services, et au niveau verticale sous la forme d'une agrégation de services.

La partie de composition horizontale de services s'appuie sur le concept de VPSN (*Virtual Private Service Network*) qui est une abstraction des ressources à provisionner pour la session de services d'un utilisateur. En effet, pour tout utilisateur, chaque service

exposable demandé est parfois composé de plusieurs éléments de services. Ces derniers sont sélectionnés et pré-provisionnés selon leurs QoS offertes (QoS prévisionnelles) dans un VPSN spécifique à l'utilisateur. Autrement dit, le VPSN n'est pas un mécanisme de réservation de ressources, mais plutôt un mécanisme de sélection et d'interconnexion virtuelle entre les éléments de services. La réservation de ressources se fait lors de la consommation sous forme d'un provisionnement dynamique selon la QoS courante des éléments de services. Dans le cadre de cette thèse, nous nous intéressons seulement à la phase de sélection qui pré-provisionne les ressources et qui est supportée par le concept VPSN.

Comme son nom l'indique, le VPSN est un "réseau" d'éléments de services. Il suit un modèle noté NLR (Nœud, Lien, Réseau) qui supporte le remplacement des serveurs d'applications centralisés par des serveurs de services distribués et interconnectés. Selon le modèle NLR, les éléments de services sont considérés comme des nœuds mutualisables qui sont enchaînés par des liens lâches selon une logique de service personnalisée afin de constituer le réseau noté VPSN. Ce réseau est "virtuel" car il s'appuie sur des éléments de services mutualisables. En effet, chaque élément de service se partage entre différents utilisateurs, et par suite entre différents VPSN. Ainsi, il n'y a pas une réservation d'une ressource service réelle pour chaque utilisateur. De plus, le VPSN s'appuie sur des liens virtuels entre ces éléments de services, ce qui rend l'architecture du VPSN de plus en plus flexible. Ainsi, le VPSN est capable de s'adapter dynamiquement et sans couture à toute modification au niveau des capacités d'un service ou au niveau du contexte ambiant et des préférences de l'utilisateur. En outre, le VPSN est "privé" car il est constitué pour répondre à une demande de service d'un utilisateur donné. Ce dernier est ainsi autorisé à accéder, tout au long de sa mobilité spatiale et temporelle, aux éléments de services sélectionnés dans son VPSN. Enfin, à l'aide de cette interconnexion horizontale virtuelle à base de VPSN, le SEIB garantit une composition de service personnalisée. Cette dernière est spécifique à chaque utilisateur et elle relie les éléments de services qui constituent les services exposables demandés et qui répondent le mieux au contexte ambiant de l'utilisateur ainsi qu'à ses préférences fonctionnelles et non-fonctionnelles (c.à.d. la QoS demandée). Cette composition personnalisée à base de VPSN est flexible et à couplages lâches, et elle s'adapte d'une manière transparente et dynamique à tout évènement temps réel.

En plus de la partie horizontale de composition de services à base de VPSN, le SEIB assure une agrégation verticale de services. Son rôle est d'interconnecter virtuellement des services appartenant à n'importe quel niveau architectural : service, réseau ou équipement. En effet, cette fonction logique introduit une vue verticale qui consiste à pré-provisionner les éléments de services, les éléments de réseaux (réseaux virtuels, systèmes autonomes, etc.) et les équipements (terminaux, points d'accès, routeurs, etc.) qui vérifient le mieux les préférences (fonctionnelles et QoS) des utilisateurs ainsi que leurs contextes ambiants. Une fois les services sont pré-provisionnés pour faire partie de la session d'un utilisateur, ce dernier sera autorisé d'y accéder tout au long de sa session. Cette agrégation verticale s'adapte d'une manière dynamique et transparente à tout changement dans les préférences de l'utilisateur ainsi que dans son contexte ambiant.

4.2.3 Les services de base

La troisième partie de notre NGN/NGS Middleware est celle des services de base. Elle est constituée d'un ensemble de composants fonctionnels développés en services tout en respectant le modèle de service introduit par notre approche SOA améliorée (voir Sous-Section 4.1.1). D'après la Figure 4.2, nous constatons, d'un point de vue architectural, une certaine analogie entre les couches de services, de réseaux et d'équipements. Cette analogie apparaît essentiellement au niveau des deux blocs architecturaux : *Mobility Management* et *Heterogeneity Management & Service Logic Personalization*. Cette structure de bout-en-bout souligne deux approches essentielles adoptées dans le cadre de notre étude. La première est l'approche *Everything as a Service* (XaaS). Elle consiste à développer toutes les fonctionnalités réseaux (routage, adressage, etc.) sous forme de services. De même, elle prend les points d'accès, les terminaux ainsi que d'autres équipements pour des sources de valeurs ajoutées vis-à-vis de l'utilisateur, et les considère ainsi comme des services. La deuxième approche soulignée par cette analogie de bout-en-bout se traduit par l'extension des principes réseaux au niveau de la couche service. Ainsi, au delà de la mutualisation qui considère le service applicatif comme une ressource à capacité partageable, nous avons fait appel à d'autres concepts réseaux comme le routage (proposition d'un routage sémantique au niveau service) et le handover (proposition d'un handover sémantique au niveau service). Grâce à cette approche de bout-en-bout, nous avons pu profiter de l'expérience réussie au niveau réseau et nous avons garanti une vision homogène de bout-en-bout pour tous les opérateurs réseaux et les fournisseurs de services et d'équipements.

D'un point de vue fonctionnel, le rôle des services de base est de lever les verrous dûs à l'émergence des contextes NGN/NGS et *user-centric*. D'une part, ils supportent la composition et l'agrégation personnalisées de services, et d'autre part, ils rendent transparent le contexte NGN sous-jacent. Dans la suite, nous présentons l'ensemble des services et des concepts qui permettent de gérer l'hétérogénéité, maintenir la continuité de services, supporter la composition personnalisée de services, respecter l'approche évènementielle, lever les verrous de sécurité et supporter une approche décisionnelle temps réel.

4.2.3.1 Modèle de QoS pour gérer l'hétérogénéité

De nos jours, l'utilisateur se trouve au centre d'un contexte NGN fortement hétérogène. Ainsi, pour composer un service global, plusieurs ressources hétérogènes doivent collaborer. Aujourd'hui, les solutions existantes prennent seulement en considération l'hétérogénéité des réseaux d'accès. Or, ce problème s'étend sur toutes les couches NGN et par suite sur toutes leurs ressources (équipements, réseaux d'accès et services). Par conséquent, le premier défi NGN à dépasser est de rendre homogène l'environnement de l'utilisateur en appliquant une perception homogène pour toutes les ressources de bout-en-bout. Pour atteindre ce but, nous proposons un modèle de QoS qui s'applique à tout type de ressource, que cela soit un équipement, un réseau d'accès ou un composant de service. Ce modèle de QoS représente le comportement et les caractéristiques de chaque ressource et représente l'aspect non fonctionnel de cette dernière. Il s'appuie

sur quatre critères génériques définis comme suit :

- Disponibilité : elle représente l'aptitude d'une ressource à être accédé à un certain moment. Elle indique le taux d'accessibilité pour chaque ressource en tenant compte des conditions contractuelles temporelles et spatiales.
- Fiabilité : elle représente l'aptitude d'une ressource à s'exécuter sans détériorer l'information traitée tout en respectant les demandes et les conditions contractuelles. Elle indique le taux involontaire de modification d'information pendant le traitement.
- Délai : il représente l'aptitude d'une ressource à s'exécuter en respectant le délai précisé dans les demandes. Il indique la durée de traitement d'une requête.
- Capacité : elle représente l'aptitude d'une ressource à traiter normalement une requête en utilisant tous les moyens de traitement possibles. Elle indique la charge maximale de chaque ressource.

Ces quatre critères de QoS sont génériques, nécessaires et suffisants. Ils s'évaluent à partir de différents paramètres mesurables qui varient selon le niveau de visibilité. Par exemple le critère de Capacité au niveau d'un équipement est représenté par sa mémoire et par la vitesse de traitement de son processeur, or dans le cas d'une ressource réseau, elle correspond au débit et à la bande passante.

Il faut noter que nous n'avons pas besoin de chercher une formulation qui combine ces quatre critères puisque toute variation d'un des critères de QoS d'une ressource sera résolue en routant la requête vers une ressource ubiquitaire (processus de gestion présenté dans le Chapitre 8).

En outre, le modèle de QoS appliqué au niveau de chaque ressource va servir comme un SLA entre l'utilisateur et le fournisseur de chaque ressource. Afin de permettre à l'utilisateur de choisir les ressources qui vérifient ses préférences non fonctionnelles et afin de supporter l'autogestion de ces ressources sélectionnées, nous définissons, pour chaque critère de QoS, différents types de valeurs :

- Valeur de conception : elle est déterminée au moment de la conception d'une ressource et elle traduit ses possibilités maximales.
- Valeur offerte : elle correspond à une adaptation de la valeur de conception au contexte de l'environnement de déploiement.
- Valeur seuil : elle indique la limite du fonctionnement normal d'une ressource et elle représente le seuil d'alerte utilisé pour déclencher les réactions adéquates et les processus d'autogestion pendant la phase d'exploitation
- Valeur courante : elle indique le comportement courant d'une ressource. Elle est surveillée tout au long de la phase d'exploitation à l'aide d'un agent de QoS qui est associé à chaque ressource. L'agent de QoS vérifie si le comportement courant de la ressource est conforme ou non à celui négocié dans le SLA. Pour cela, il compare la valeur courante à la valeur seuil à ne pas dépasser. Si la valeur courante est inférieure à la valeur seuil, il envoie un *OUT-Contract* afin de lancer le processus de gestion de mobilité pour empêcher la coupure de la session. Sinon, il envoie un *IN-Contract*.

4.2.3.2 Services de base pour personnaliser la logique de services

Suite à notre approche *user-centric*, la personnalisation de services devient un défi majeur du contexte NGS. Afin de lever ce verrou, nous avons proposé une interconnexion virtuelle horizontale dénommée VPSN qui pré-provisionne les éléments de services répondant le mieux aux besoins de l'utilisateur, à son contexte ambiant et à ses préférences en temps réel. Par conséquent, chaque utilisateur est capable de composer sa propre session de services personnalisée à partir des éléments de services pré-provisionnés dans son VPSN. Afin de supporter cette composition de services personnalisée à base de VPSN, nous proposons les services suivants :

- *Service Logic* : il décrit l'enchaînement logique des éléments de services applicatifs afin de représenter la composition des services globaux demandés par l'utilisateur.
- *Semantic Routing* : il route la requête vers le serveur hébergeant l'élément de services suivant dans la logique de services de l'utilisateur. Ce routage est sémantique puisqu'il prend en considération l'enchaînement et les préférences fonctionnelles de l'utilisateur en routant selon la logique de services demandée par ce dernier. Cependant, afin de prendre encore en considération les préférences non fonctionnelles de l'utilisateur, nous supportons le routage sémantique par une table de routage basée QoS. Ainsi, chaque service S_i dans le VPSN stocke dans sa table de QoS les informations suivantes : la QoS des services qui lui sont ubiquitaires (concept de communautés virtuelles introduit dans la Partie II lors de la gestion de mobilité) ainsi que la QoS des liens qui connecte S_i à ces différents services. Ensuite, il classifie les services ubiquitaires dans sa table de QoS du plus proche au plus loin d'un point de vue réseau. Pour ce faire, il calcule une QoS_{totale} qui est la somme de sa propre QoS avec la QoS du service ubiquitaire ainsi que la QoS du lien qui les relie. Par conséquent, dans le cas de la dégradation du service S_i, il pourra router les requêtes qu'il lui arrive vers le premier service ubiquitaire dans sa table de QoS. Ce dernier est le plus adéquat pour remplacer S_i dans la logique de services et il est censé de pourvoir répondre à cette requête. Ainsi, d'après le routage sémantique, nous dépassons la coupure de la session en routant d'une manière dynamique et transparent vers un autre service ubiquitaire.
- *Service Selection* : Il sélectionne le service choisi pour faire partie de la logique de services.

Selon le routage sémantique supporté par la table de routage à base de QoS, le service (nœud) n'est pas le seul élément primordial dans la composition de services, il y a aussi les liens qui doivent être pris en considération. En effet, la transition d'un élément de service vers un autre passe obligatoirement par un réseau de transport qui s'appuie sur des ressources réseaux et sur des équipements. Ainsi, par analogie au VPSN créé au niveau service, nous pré-provisionnons aussi les ressources réseaux (systèmes autonomes, réseaux virtuels, etc.) et les équipements (routeurs, terminaux, etc.) nécessaires pour composer la logique de services, dans un VPCN (*Virtual Private Connectivity Network*) et dans un VPEN (*Virtual Private Equipment Network*) respectivement. Afin de supporter ces VPCN et ces VPEN, nous proposons par analogie au niveau service, les composants suivants : le *Network Logic* et l'*Equipment Logic* qui

indiquent respectivement l'ensemble des composants réseaux et équipements qui seront utilisés dans la session de l'utilisateur ; le *Network Routing* et *Equipment Routing* qui correspondent aux techniques de routage existantes (ex. BGP, MPLS, OSPF, etc.) qui seront eux-mêmes développés en services ; et enfin le *Network Selection* et l'*Equipment Selection* qui sélectionnent les ressources réseaux et les équipements choisis selon les techniques de routage.

4.2.3.3 Services de base pour gérer la mobilité

Dans un contexte *user-centric*, la mobilité est un des verrous primordiaux à lever. Afin de gérer la mobilité, nous proposons deux nouveaux concepts : le handover sémantique et les communautés virtuelles. Comme le montre la Figure 4.2, nous adoptons une certaine analogie entre les différentes couches NGN afin d'assurer une gestion cohérente de bout-en-bout. En effet, le bloc fonctionnel pour la gestion de mobilité à l'aide du handover sémantique présente, au niveau service, une certaine analogie par rapport aux handovers qui existent au niveau des réseaux d'accès et du réseau cœur. Ces derniers seront aussi représentés sous forme de services selon notre approche *Everything as a Service*. Quand à la gestion de la mobilité à l'aide des communautés virtuelles, elle ne correspond pas à une extension d'une solution existante au niveau réseau, mais elle correspond plutôt à un nouveau concept qui se base sur la notion d'ubiquité et qui est appliqué sur les différentes couches NGN (service, réseau et équipement). Comme le montre la Figure 4.2, nous proposons au niveau service, un bloc fonctionnel pour la gestion des VSCUs (*Ubiquity-based Virtual Service Communities*). Chaque VSCU correspond à une communauté virtuelle qui associe un ensemble de services ubiquitaires, c.à.d. ayant la même fonctionnalité et une QoS équivalente. Par analogie, nous avons aussi conçu, les VCCUs (*Ubiquity-based Virtual Connectivity Communities*) et les VECUs (*Ubiquity-based Virtual Equipment Communities*) qui créent respectivement des communautés virtuelles au niveau des connectivités réseaux et au niveau des équipements.

En effet, cette partie de gestion de la mobilité à l'aide du handover sémantique et des communautés virtuelles sera plus détaillée dans le Chapitre 8, dans lequel nous nous focalisons sur le niveau service. Pour cela, en ce qui concerne la gestion des communautés virtuelles, nous traitons seulement le cas des VSCUs (les VCCUs et les VECUs seront traitées par analogie).

Cependant, afin de supporter nos propositions de gestion de la mobilité, nous définissons un ensemble de services de base décrits comme suit :

- LBS (*Location Basic Service*) : il récupère de la base de connaissances l'adresse logique (SIP URI) d'une ressource en ayant comme entrée l'identifiant de cette dernière.
- Geo-LBS (*Geographic-Location Basic Service*) : il récupère de la base de connaissances la position géographique (longitude, latitude) d'une ressource en ayant comme entrée l'identifiant de cette dernière.
- FBS (*Find Basic Service*) : il cherche dans la base de connaissances, pour un rôle (service, réseau ou équipement) déterminé et selon certains critères demandés,

toutes les ressources qui vérifient ce rôle ainsi que l'ensemble de ces critères. Pour ce faire, il reçoit comme entrée le rôle des ressources à chercher ainsi que les critères de cette recherche. Ainsi, il donne comme sortie la liste des identifiants des ressources trouvées.

- DBS (*Discovery Basic Service*) : il découvre dans un réseau pair-à-pair JXTA (*Juxtapose*) [62], pour un rôle (service, réseau ou équipement) déterminé et selon certains critères demandés, toutes les ressources qui vérifient ce rôle ainsi que l'ensemble de ces critères. Pour ce faire, il reçoit comme entrée le rôle des ressources à découvrir dans le réseau JXTA ainsi que les critères de cette découverte. Ainsi, il donne comme sortie la liste des identifiants des ressources trouvées ainsi que leurs adresses logiques (SIP URIs). D'autres informations peuvent être aussi fournies par le service DBS comme par exemple la zone de couverture (utilisée dans la suite pour le handover sémantique) et l'identifiant du fournisseur de chacun des services découverts.
- PBS (*Presence Basic Service*) : il récupère de la base de connaissances l'état d'une ressource en ayant comme entrée l'identifiant de cette dernière. Il permet de savoir si une ressource est "Disponible", "Activable" ou "Activée". En effet, une ressource est "Disponible" quand elle est accessible pour l'utilisateur. Elle est "Activable" quand elle est accessible pour l'utilisateur et qu'elle possède les capacités suffisantes pour supporter cet utilisateur et tous ceux qui l'ont déjà sélectionnée dans leur VPxN (x=S pour la couche service, x=C pour la couche réseau, x=E pour la couche des équipements). Enfin, une ressource est "Activée" quand elle est en train d'être utilisée, et elle fait partie d'une transaction temps réel dans la session de l'utilisateur.
- UBS (*Ubiquity Basic Service*) : il récupère de la base de connaissances la fonctionnalité et les quatre critères de QoS courante d'une ressource, en ayant comme entrée l'identifiant de cette dernière.
- *VPSN Maintenance* : il maintient le VPSN de l'utilisateur en l'adaptant à tout changement décidé par les autres services de gestion. Par analogie, il y a aussi le *VPCN Maintenance* et le *VPEN Maintenance* qui maintiennent respectivement le VPCN et le VPEN de l'utilisateur.

4.2.3.4 Autres services de base

Afin de supporter notre approche évènementielle, nous proposons un gestionnaire d'évènements (*Events Manager*). Ce dernier associe à chaque évènement une action qui consiste à notifier tous les composants de l'architecture qui ont souscrit à ce type d'évènement. Cet *Events Manager* n'est pas un simple aiguilleur, il effectue aussi un traitement plus complexe qui consiste à corréler une suite d'évènements pendant une certaine période afin de prendre une décision plus intelligente et par suite lancer un processus spécifique pour gérer l'ensemble de ces évènements.

Nous trouvons aussi dans notre NGN/NGS Middleware, des services de base qui existent normalement dans d'autres architectures de services comme les services IT, la tarification, etc. De plus, dans le cadre de notre projet UBIS, notre équipe de recherche

s'est intéressée aussi aux problématiques de sécurité, et elle a ainsi proposé un *Securityware* qui correspond à une plate-forme contenant un ensemble de services de base qui traitent l'authentification, l'autorisation, la fédération et l'audit. Ce *Securityware* est basé sur une approche "*Single Sign-On*" qui permet de gérer l'authentification unique, non seulement auprès des composants de services applicatifs mais aussi au niveau de l'ensemble des terminaux qui forment le réseau ambiant personnel de l'utilisateur. La proposition s'appuie sur un ensemble de *Policy Agents* qui sont introduits dans les terminaux, considérés comme des plates-formes, et dans les plates-formes de services. Ces *Policy Agents* contactent le *Securityware* qui permet d'une part de contrôler, selon le rôle de l'utilisateur, ses droits au niveau de l'usage des terminaux faisant partie de son réseau ambiant personnel, et d'autre part de vérifier la correspondance des droits de cet utilisateur et les permissions des éléments de service demandés.

En outre, autres que ces services de base à valeurs ajoutées, il y a aussi, une interaction entre une grille ambiante dénommée *AmbientGrid* et une base de connaissances, dénommée *Infoware* [57]. Cette partie, elle aussi, n'est pas dans le cadre de cette thèse mais elle fait partie du projet UBIS. En effet, l'*Infoware* qui est une base de connaissances structurée, gère efficacement et d'une manière dynamique les informations décisionnelles et réactives. Ce n'est pas une simple base de données comme est le cas du HSS (*Home Subscriber Server*) [2] dans l'architecture IMS. Au contraire, c'est une base d'informations qui agit comme une inférence informationnelle. Comme nous le constatons dans la Figure 4.2, l'*Infoware* est composé de plusieurs profils XML (*Service Profile, Real-time Profile, VPSN Profile, VSCU Profile*, etc.) et supporte un méta-modèle informationnel bien défini qui tient compte des quatre niveaux de visibilité (utilisateur, service, réseau et équipement) et qui organise toutes les informations nécessaires tout en adoptant une vision de QoS, afin d'avoir partout une adaptation dynamique de service. Selon notre approche distribuée et trans-organisationnelle, non seulement les services sont distribués, mais aussi les informations décisionnelles et les profils XML sont distribués entre différents *Infoware*. Pour cela, nous adoptons une approche de fragmentation des *Infoware* et nous considérons que chaque fournisseur possède un fragment d'*Infoware* associé à sa plate-forme. Ce fragment contient seulement les informations qui sont sous la responsabilité de la plate-forme et qui sont nécessaires à cette dernière afin d'effectuer les traitements de ses services. Quand aux informations qui sont communes pour un ensemble de fournisseurs, elles sont partagées entre les différents fragments d'*Infoware* et sont mises à jour dynamiquement.

L'autre partie de l'interaction est l'*AmbientGrid* qui est une sorte d'inférence qui dépend fortement des profils définis dans l'*Infoware*, et qui permet à l'utilisateur de gérer ses ressources ambiantes d'une manière personnalisée et dynamique. L'*AmbientGrid* donne les meilleurs choix au niveau terminal, réseau et service, afin que l'utilisateur puisse composer sa propre session selon ses propres préférences. Pour ce faire, l'*AmbientGrid* utilise comme entrée "les *Active Profiles*" qui représentent l'ensemble des ressources ambiantes qui sont 'Disponibles", "Activables" et "Activées", et le *User Profile* qui précise les préférences de l'utilisateur.

5 Conclusion

Dans cette partie, nous avons proposé une nouvelle architecture de service, dénommée NGN/NGS Middleware, qui dépasse les problématiques des architectures Telco monolithiques, les défis des architectures Web favorisant l'approche client/serveur surtout avec l'émergence du Web 2.0 à base du protocole REST, et enfin les limitations des architectures SOA existantes surtout au niveau de la conception des services et au niveau de la gestion des besoins des utilisateurs, de leurs préférences, de leurs mobilités et de leurs contextes ambiants. Pour ce faire, notre architecture suit un modèle architectural horizontal, distribué et trans-organisationnel sous forme de *Middleware* de services. Elle structure selon une approche EDA et SOA améliorée, l'ensemble des services destinés pour lever les verrous NGN/NGS dans un contexte *user-centric*. Ces services suivent un modèle de service favorisant l'autogestion, la mutualisation et l'interconnexion par des liens génériques à couplages lâches. Ainsi, ce modèle de service supporte la convergence des services en s'appliquant à tout type de services Telco, Web ou IT. De plus, il aide à avoir une session de service personnalisée qui peut s'adapter dynamiquement et d'une manière transparente à tout changement dans les préférences de l'utilisateur et dans son contexte ambiant.

Afin de supporter d'une part, cette personnalisation et convergence de services, et d'autre part, la cohabitation avec le contexte NGN sous-jacent, notre NGN/NGS Middleware introduit un ensemble de concept et de services structurer en trois partie : les services exposables, le SEIB et les services de base. Comme leur nom indique, les services exposables seront exposés dans les catalogues des fournisseurs. Le SEIB est un ESB amélioré qui assure en plus des fonctionnalités d'interopérabilité, de médiation et de gestion des fils d'attente, une nouvelle fonctionnalité d'interconnexion virtuelle qui répond au besoin de transparence vis-à-vis de l'utilisateur et qui assure la personnalisation de service. Cette fonctionnalité se manifeste par une interconnexion horizontale sous forme d'un VPSN et par une interconnexion verticale sous forme d'une agrégation de services de bout-en-bout. En effet, ces deux types d'interconnexions virtuelles se basent sur un pré-provisionnement des ressources qui répondent le mieux aux besoins de l'utilisateur et auxquels ce dernier est autorisé à accéder tout au long de sa session. Cette composition horizontale et cette agrégation verticale de services sont

adaptées en temps réel et d'une manière dynamique et transparente à tout changement de préférences de l'utilisateur ou de contexte ambiant. Afin de supporter cette personnalisation de services, des services de base ont été proposés. Ils permettent de composer la logique de services demandée par l'utilisateur tout en tenant compte non seulement des préférences fonctionnelles de ce dernier, mais aussi de ses préférences non fonctionnelles, c.à.d. la QoS demandée. Pour supporter cette approche, un modèle de QoS a été proposé. Il est basé sur quatre critères génériques, nécessaires et suffisants : la disponibilité, la fiabilité, le délai et la capacité. En l'appliquant sur toutes les ressources, ce modèle permet ainsi d'homogénéiser l'environnement de l'utilisateur. Par conséquent, nous avons pu dépasser le problème de l'hétérogénéité qui constitue un des défis majeurs du contexte NGN. Le deuxième défi levé à l'aide des services de base est celui de la mobilité. Cependant, cette partie ne détaille pas les services de base qui répondent aux besoins de mobilité spatiale et temporelle de l'utilisateur.

Il faut noter que notre architecture s'appuie sur un cœur IMS [2]. Cette couche sous-jacente contrôle la session IP ainsi que la livraison du flux média, mais elle n'est plus responsable de l'interaction et de l'orchestration entre les serveurs d'applications, comme c'était le cas avec le serveur d'interaction SCIM. En effet, c'est le rôle du NGN/NGS Middleware de sélectionner et de pré-provisionner les services d'une manière optimale et de les provisionner dynamiquement lors de l'exploitation selon une chorégraphie à couplage lâche qui se traduit par une composition dynamique de services. Il faut aussi noter que l'*Infoware* est une amélioration majeure de la base HSS proposée par l'architecture IMS. D'autre part, la dimension protocolaire s'est aussi adaptée à cette évolution au niveau architectural. Pour cette raison, le protocole SIP+ est proposé. Ce dernier ne fait pas partie de cette thèse, mais fait partie de notre projet de recherche UBIS. Ce protocole est une amélioration du protocole SIP. Il assure d'une part l'interaction entre le cœur IMS et les services, et d'autre part, l'interaction entre les services eux-mêmes. Dans les deux cas, les messages SIP+ transportent des informations de QoS, ce qui garantit ainsi l'établissement de la session de l'utilisateur tout en tenant compte de la QoS demandée de bout-en-bout.

Dans la partie suivante, nous détaillons le bloc de gestion de la mobilité, qui constitue un bloc fonctionnel primordial dans la partie des services de base de notre NGN/NGS Middleware. Le but est de garantir pour chaque utilisateur une continuité de services tout au long de sa mobilité spatiale et temporelle, sans avoir négligé le changement temps réel dans son contexte ambiant.

Deuxième partie

Gestion de la mobilité

6 Introduction

Ayant défini notre architecture NGN/NGS Middleware, nous détaillons dans cette partie, le bloc fonctionnel qui gère la mobilité. En effet, la gestion de la mobilité est un des piliers du contexte *user-centric* dans lequel chaque utilisateur désire établir une session unique et continue, qui prend en considération son contexte ambiant ainsi que ses préférences fonctionnelles et sa QoS de bout-en-bout demandée. Afin de garantir la continuité de la session de bout-en-bout, quatre types de mobilité doivent être gérés : la mobilité de l'utilisateur, la mobilité du terminal, la mobilité du réseau et la mobilité de services. Il faut noter que dans le cadre de cette thèse, seule la mobilité du réseau n'est pas prise en considération.

De nos jours, l'émergence des contextes NGN et NGS a favorisé de plus en plus l'approche *user-centric* et a fortement impacté le défi de mobilité. D'une part, d'après le contexte NGS, chaque utilisateur possède une session de services personnalisée qui est composée par des éléments de services répondant le mieux à ses préférences fonctionnelles et non fonctionnelles. Or, les services de nouvelle génération sont mutualisables, ce qui implique un pré-provisionnement virtuel des ressources service, qui s'appuie sur un partage de chaque ressource service entre différents utilisateurs (ou autrement dit, entre différents VPSN). À cause de ce partage de ressource, le service auquel l'utilisateur est abonné peut avoir une QoS courante qui se révèle insuffisante par rapport à celle négociée dans le contrat de SLA. Afin d'anticiper ce non respect des préférences de l'utilisateur, nous avons besoin d'une solution qui gère d'une manière dynamique et transparente la mobilité de la session dans le cas d'une dégradation de QoS ou d'un mauvais fonctionnement d'un service pré-provisionné.

D'autre part, l'émergence du contexte NGN et l'évolution rapide des technologies au niveau des réseaux d'accès et des terminaux intelligents, stimulent les utilisateurs d'aujourd'hui à être de plus en plus nomades. Cependant, chaque mobilité spatiale (mobilité de l'utilisateur ou mobilité du terminal) est accompagnée par un changement au niveau du contexte ambiant de l'utilisateur. Ce dernier, selon notre approche *user-centric*, désire être de plus en plus conscient de son contexte ambiant. En effet, le fait d'avoir conscience et par suite d'avoir accès aux ressources ambiantes assure beaucoup d'avantages vis-à-vis de l'utilisateur : ça optimise la QoS de bout-en-bout,

diminue la latence, améliore la QoE de l'utilisateur et permet à ce dernier d'avoir différentes préférences dans différents contextes (bureau, maison, vacances, etc.) et d'avoir plusieurs possibilités d'accès aux services liées à la localisation, à la capacité de l'équipement ou au type d'opérateur. Afin d'améliorer la prise en conscience du nouveau contexte ambiant de l'utilisateur et afin de traiter de façon plus efficace les nouvelles informations ambiantes, le concept des réseaux ambiants a été conçu. Cependant, cette solution reste toujours limitée aux réseaux et aux terminaux, et elle néglige toute prise en conscience du contexte de l'utilisateur au niveau de la couche service. Pour dépasser ce problème, nous avons besoin d'une solution qui intervient au niveau de la couche service et qui adapte, d'une manière continue dynamique et transparente, la session de services (VPSN) de l'utilisateur à tout changement au niveau de son contexte ambiant.

Nous commençons cette partie par une analyse des solutions existantes qui traitent la gestion de la mobilité sur les différentes couches NGN (voir Chapitre 7). Après avoir discuté leurs avantages et leurs défaillances, nous proposons dans le Chapitre 8 nos deux nouveaux concepts pour la gestion de la mobilité, permettant de lever les deux derniers verrous identifiés dans le Chapitre 1. Le premier est celui des communautés virtuelles (voir Section 8.1) qui est basé sur le principe d'ubiquité et qui traite la continuité de la session de bout-en-bout, suite à un mauvais fonctionnement ou une dégradation de QoS de la part d'une des ressources participantes à la session de l'utilisateur. Le deuxième concept est celui du handover sémantique (voir Section 8.2) qui gère, tout au long de la mobilité temporelle de l'utilisateur, la continuité de services suite à un changement dans le contexte ambiant de l'utilisateur dû à une mobilité spatiale. À la fin, nous concluons cette partie (voir Chapitre 9).

7 Analyse des solutions existantes pour la gestion de la mobilité

Au cours de ces dernières années, plusieurs solutions ont évolué afin de dépasser le défi de mobilité. Dans ce chapitre, nous présentons l'ensemble des techniques qui avaient été conçues pour gérer la mobilité au niveau des réseaux d'accès et au niveau du réseau cœur IP. Au début, nous discutons les mécanismes de handover horizontal et vertical qui sont adoptés pour la gestion de la mobilité au niveau des réseaux d'accès. Ensuite, nous discutons le concept de MIP (*Mobile IP*) qui gère la mobilité de l'adresse IP de l'utilisateur. Il faut noter que pour chacune de ces techniques de gestion de mobilité, nous identifions trois rôles essentiels : l'initiateur, le décideur et l'exécuteur. Cependant, plusieurs acteurs peuvent collaborer afin de jouer chacun de ces rôles.

7.1 Handover horizontal au niveau des réseaux d'accès

Le mécanisme de handover horizontal permet de gérer la mobilité au sein d'une même génération de réseau d'accès cellulaire. Dans ce qui suit, nous présentons l'ensemble des solutions existantes qui adopte ce mécanisme de handover horizontal.

7.1.1 Réseaux cellulaires 2G (GSM) et 2.5G (GPRS/EDGE)

En 1990, l'ETSI a publié la première phase des spécifications du réseau cellulaire de deuxième génération, dénommé GSM (*Global System for Mobile communications*). Ce dernier est devenu le réseau mobile le plus répandu dans le monde entier. Afin de prendre en considération la transmission par paquets, l'ETSI a publié en 1999 la première phase d'une nouvelle technologie, dénommée GPRS (*General Packet Radio Service*) qui est vue comme une extension du GSM. En 2000, la publication de la première phase d'un GPRS amélioré, dénommé EDGE (*Enhanced Data Rates for GSM Evolution*), a permis d'avoir des débits beaucoup plus élevés au niveau des canaux radio des réseaux cellulaires.

Au niveau de la gestion de la mobilité, ces trois types de réseaux cellulaires maintenus de nos jours par le 3GPP, interviennent d'une manière équivalente [17][35][23] [20]. Ils s'appuient sur un mécanisme de handover horizontal basé sur trois acteurs essentiels : le MS (*Mobile Station*), le BSC (*Basic Station Controller*) et le MSC (*Mobile Switching Center*) :

- Le MS joue le rôle d'initiateur du handover. Il effectue des mesures (le niveau de signal reçu noté *RXLEV*, la qualité du signal notée *RXQUAL* et le taux d'erreur bit noté *BER*) sur le canal courant et sur les six cellules voisines ayant les niveaux de puissance les plus élevés. Ces mesures sont ensuite transmises au BSC courant, ce qui lui permet d'établir une liste de cellules candidates pour le handover.
- Ainsi, le BSC peut soit jouer le rôle de décideur du handover (cas du handover intra-BSC) soit de laisser ce rôle au MSC (cas du handover inter-BSC). Pour le premier cas, le BSC choisit, parmi une liste de cellules candidates, la cellule cible la mieux adaptée pour des raisons entre autre de trafic. Pour le deuxième cas, c'est le MSC qui fait se traitement. Il faut noter, qu'un handover inter-MSC peut aussi s'effectuer. Pour le gérer, un *MSC Ancre* joue le rôle de décideur.
- Enfin, le BSC joue le rôle d'exécuteur du handover. Il reçoit la décision du décideur, réserve un nouveau canal de trafic entre la cellule cible et le MS et demande au MS de basculer sur ce canal.

Nous constatons d'après ce processus de handover, que ces trois technologies (GSM, GPRS et EDGE) interviennent seulement au niveau des réseaux d'accès et ne traitent que la mobilité du terminal.

7.1.2 Réseaux cellulaires 3G (UMTS)

En 1999, le 3GPP a publié le *Release'99* de la troisième génération des réseaux cellulaires, dénommée UMTS (*Universal Mobile Telecommunications System*). Cette dernière repose sur une approche d'étalement de spectre et sur une technologie d'accès multiple, dénommée W-CDMA (*Wideband Code Division Multiple Access*). Afin de gérer la mobilité, l'UMTS se base sur un mécanisme de handover horizontal [18][19] qui se manifeste selon deux approches : le *Soft-Handover* et le *Hard-Handover*. Ces deux techniques s'appuient sur trois acteurs essentiels : l'UE (*User Equipment*), le Node-B et le RNC (*Radio Network Controller*).

D'une part, le *Soft-Handover* est basé sur le concept de *make-before-break* qui consiste à établir une nouvelle connexion avant de couper l'ancienne. Ce *Soft-Handover* s'effectue lorsque l'UE est dans une zone de couverture supportée par différents Node-B constituant son *Active Set*. L'UE transmet et reçoit simultanément des données de ces différents Node-B. Les données émises par l'UE sont reçues par chaque Node-B qui les transmet au RNC où elles sont combinées, quand aux données émises simultanément par chaque Node-B, elles sont combinées dans l'UE. Lorsque les Node-B sont contrôlés par différents RNC, c'est le *Serving-RNC* qui combinent les données utilisateur. La technique de *Soft-Handover* est décrite selon le processus suivant :

- L'UE mesure les *Received Signal Code Power* et les *Received Signal Strength Indicator* et renvoie ces informations vers les Node-B de son (*Active Set*). Ensuite,

les Node-B renvoient les mesures vers le *Serving-RNC* (nous prenons le cas où il y a plusieurs RNC). Ce dernier compare les puissances de signaux mesurées à un certain seuil, noté As_Th. Suite à cette comparaison, le *Serving-RNC* initie le handover et notifie l'UE.

- Ainsi, l'UE joue le rôle de décideur en décidant d'ajouter (*Radio Link Addition*), de relever (*Radio Link Removal*) ou de remplacer (*Combined Radio Link Addition and Removal*) un Node-B dans l'*Active Set*.
- Enfin, le *Serving-RNC* joue le rôle d'exécuteur du handover. Il met à jour l'*Active Set* et envoie vers l'UE le message de signalisation *Active Cell Update Complete*.

D'autre part, le *Hard-Handover* est basé sur le concept de *break-before-make* qui consiste à couper l'ancienne connexion avant d'établir la nouvelle. Normalement, quand l'UE passe dans une cellule où les fréquences diffèrent de celles qu'il vient de quitter, un *Hard-Handover* s'effectue. Nous parlons ainsi d'un handover inter-fréquences. Dans le cas de ce *Hard-Handover*, le *Serving-RNC* initie le mécanisme de handover en se basant sur les mesures radio rapportées par l'UE. Ensuite, c'est aussi le *Serving-RNC* qui joue le rôle de décideur et qui choisit le Node-B cible. Enfin, le *Serving-RNC* exécute le handover en libérant le lien radio entre l'UE et l'ancien Node-B, avant d'établir le lien entre cet UE et le Node-B cible.

Nous constatons que comme dans le cas des réseaux mobile 2G, l'UMTS n'intervient qu'au niveau accès et ne traite que la mobilité du terminal.

7.1.3 Réseaux cellulaires 4G (LTE)

Depuis l'année 2006, le 3GPP commence à mettre en œuvre ses études d'amélioration des réseaux 3G existants, en spécifiant une technologie de quatrième génération, dénommée LTE (*Long Term Evolution*). Au niveau de la gestion de mobilité, le LTE adopte une approche de handover horizontal [22] qui s'appuie sur quatre acteurs essentiels qui sont l'UE, le eNode-B source, l'eNode-B cible et le MME (*Mobility Management Entity*). Afin d'initier, décider et exécuter le handover horizontal, ces acteurs collaborent de la façon suivante :

- L'UE initie le handover en envoyant à l'eNode-B source un rapport de mesures qui correspondent aux puissances des signaux des eNode-B candidats.
- L'eNode-B source joue le rôle de décideur. Il s'appuie sur le rapport reçu afin de choisir l'eNode-B cible le plus adéquat. Ensuite, il envoie une requête de handover (*HO Request*) vers l'eNode-B cible.
- Enfin, l'exécution de ce handover se base sur une interaction entre les différents acteurs. En effet, après avoir reçu le *HO Request*, l'eNode-B cible enregistre le contexte, prépare les couches basses pour le handover et répond au eNode-B source par un acquittement (*HO Request Ack*). L'eNode-B source transmet toutes les informations à l'UE. Ensuite, il arrête d'échanger des messages avec lui et fait suivre tous les messages provenant de l'UE à l'eNode-B cible. L'UE informe l'eNode-B cible du succès du handover avec un message de confirmation. Ainsi, l'eNode-B cible initie le changement de chemin de données en envoyant un *Path Switch* au MME pour l'informer que l'UE a changé de cellules. Le MME en-

voie ainsi un message *Update User Plane Request* vers un *Serving Gateway* qui se charge de changer le chemin des données en *Downlink* pour qu'ils atteignent l'eNode-B cible. Ensuite, le *Serving Gateway* envoie un *Update User Plane Response* au MME qui envoie à son tour un *Path Switch Acknowledge* vers l'eNode-B cible. A la réception de ce message, l'eNode-B cible envoie une indication *UE Context Release* à l'eNode-B source pour qu'il libère définitivement la connexion avec l'UE.

Nous constatons que comme pour les autres générations de réseaux cellulaires, le LTE gère essentiellement la mobilité du terminal au niveau accès.

7.1.4 WiMAX

En parallèle à l'avancement au niveau LTE, une autre technologie se déclare comme candidate pour le titre de réseau cellulaire de quatrième génération. Cette technologie est l'IEEE 802.16, dénommé WiMAX (*Worldwide Interoperability for Microwave Access*). Au niveau de la gestion de mobilité, l'IEEE 802.16e [59] définit trois types de handover : le *Hard-Handover*, le MDHO (*Micro Diversity Handover*) et le FBSS (*Fast Base Station Switching*). Le *Hard-Handover* s'appuie, comme dans l'UMTS, sur la technique de *Break Before Make*. C'est le cas quand le MS est relié à une seule BS (*Base Station*). Selon le *Hard-Handover*, les connexions avec le *Serving-BS* sont libérées avant que le MS soit connecté à un *Target-BS*. Les deux autres types (MDHO et FBSS) sont considérés comme des *Soft-Handovers* basés sur le principe de *Make Before Break*. Pour ces deux cas, le MS est connecté simultanément à un ensemble de BS, dénommé *Diversity Set*. Dans le cas du MDHO, le MS communique simultanément avec toutes les BSs du *Diversity Set*. Il effectue un *Diversity Combining* pour les signaux *Downlink* provenant des différentes BS. En outre, chaque BS effectue un *Selection Diversity* sur les signaux *Uplink* provenant du MS. Dans le cas du FBSS, le MS communique seulement avec un *Anchor-BS* qui fait partie du *Diversity Set*.

Au niveau du processus de handover, les trois types de handover s'appuient sur une approche à trois rôles (initiateur, décideur et exécuteur). En effet, le *Hard-Handover* dans WiMAX ressemble à celui appliqué dans le 3G, mais avec des acteurs différents. De même, le MDHO dans WiMAX ressemble au *Soft-Handover* appliqué dans le 3G, mais avec des acteurs différents et avec un *Diversity Set* au lieu de l'*Active Set*. Pour cette raison, dans ce qui suit, nous explicitons le processus de handover dans le cas de FBSS. Afin de mettre en œuvre les trois rôles, trois acteurs interviennent : le MS, le *Serving-Anchor-BS*, et le *Target-Anchor-BS* :

- Le MS joue le rôle d'initiateur. Il examine son entourage dans des *Scanning Intervals* et fait des mesures sur les canaux de *Downlink* reçus de la part de son *Diversity Set*. Il mesure le *Carrier to Interference-plus-Noise Ratio*, le *Receive Signal Strength Indicator*), le *Relative Delay* et le *Round Trip Delay*. Ensuite, les mesures faites sont transmises au *Serving-Anchor-BS*.
- Ainsi, le *Serving-Anchor-BS* joue le rôle de décideur. Il choisit le meilleur *Target-Anchor-BS* en se basant sur les paramètres reçus, et puis il le notifie.
- Enfin, le *Target-Anchor-BS* sélectionné exécute le handover en ouvrant des ca-

naux de communications avec le MS. Une fois la connexion établie entre le *Target-Anchor-BS* et le MS, la connexion peut être relâchée entre ce MS et l'ancien *Serving-Anchor-BS*.

Pour conclure, nous constatons que le WiMAX gère, lui aussi, la mobilité du terminal au niveau des réseaux d'accès.

7.2 Handover vertical entre les réseaux d'accès

À cause de la diversité des technologies et leur inévitable cohabitation, nous avons besoin d'un mécanisme de handover qui gère la mobilité entre des milieux hétérogènes. Pour cela, le mécanisme de handover vertical est conçu. Dans ce qui suit, nous présentons l'ensemble des solutions existantes qui adopte ce mécanisme de handover vertical.

7.2.1 UMA/GAN

UMA/GAN (*Unlicensed Mobile Access/Generic Access Network*) assure la continuité des communications entre un réseau GSM/GPRS/UMTS et un réseau local sans fil tel le WiFi ou le *Bluetooth* et inversement [21]. Il s'agit ici d'utiliser un terminal bi-mode qui se trouve dans une zone de couverture GAN et qui peut fonctionner avec une licence dans un réseau mobile GSM, GPRS ou UMTS, et sans licence dans un réseau local Wifi ou *Bluetooth*. Afin d'expliquer le processus de gestion de mobilité par UMA/GAN, nous prenons l'exemple d'un handover vertical entre un réseau GSM et un réseau WiFi. Le processus de handover [5] respecte l'approche des trois rôles (initiateur, décideur et exécuteur) effectués par plusieurs acteurs. Il est décrit comme suit :

- Le MS joue le rôle d'initiateur du handover. Il envoie un rapport de mesures au BSC, tout en indiquant les caractéristiques des cellules voisines ayant les puissances de signaux les plus élevées, y compris celles de la cellule UMA.
- Ensuite, le BSC joue le rôle de décideur. Ainsi, après avoir analysé le rapport, il décide de choisir la cellule UMA comme cellule cible, et il envoie à son GSM-CN (*GSM-Core Network*) un message *Handover Required* tout en indiquant sa décision. Le GSM-CN demande à l'UmaNC (*UMA Network Controller*), aussi noté GANC (*GAN Network Controller*), d'allouer un nouveau canal radio au mobile. Une fois l'UmaNC accepte ce handover, il informe le MS par l'intermédiaire du réseau GSM.
- Enfin, pour la phase d'exécution, le MS contacte l'UmaNC pour que ce dernier lui alloue les ressources nécessaires en mettant en place un tunnel de communication. Ensuite, l'UmaNC indique au GSM-CN qu'il est bien en communication avec le MS. Ainsi, le GSM-CN libère les ressources allouées au BSC qui à son tour confirme la libération des ressources allouées au MS.

Malgré sa capacité à tenir compte de ce contexte hétérogène, le UMA/GAN est toujours limité à une gestion de mobilité du terminal au niveau des réseaux d'accès.

7.2.2 MIH

Le standard IEEE 802.21 [58], dénommé MIH (*Media Independent Handover*), est en cours de développement. Il élabore une architecture qui permet le passage sans coupure d'un réseau homogène et/ou hétérogène à un autre. Pour ce faire, MIH crée une nouvelle fonction, notée MIHF, qui est constituée de trois services :

- Le MIES (*Media Independent Event Service*) : il génère des évènements qui peuvent indiquer ou prévoir un changement dans l'état ou le comportement de transmission des couches physique et liaison. Ces évènements sont transmis d'une couche basse à une couche haute dans laquelle la décision sera prise.
- Le MICS (*Media Independent Command Service*) : il définit les décisions prises par les couches supérieures afin de contrôler les couches basses. En effet, ces décisions permettent entre autre de déterminer l'état des liens, de contrôler les performances du terminal multi-mode, et de renseigner dynamiquement sur le débit du lien, la puissance du signal ainsi que d'autres informations décisionnelles.
- Le MIIS (*Media Independent Information Service*) : il offre la possibilité de découvrir les réseaux entourant le MS et de sélectionner le réseau adéquat.

En effet, le MIH n'exécute pas le handover ; il aide simplement la couche supérieure à exécuter les handovers des couches basses, en envoyant des informations décisionnelles.

7.3 Handover au niveau du réseau cœur (Mobile IP)

Par analogie aux réseaux d'accès, le réseau cœur IP nécessite, lui aussi, un mécanisme de handover pour gérer la mobilité. Pour cette raison, l'IETF (*Internet Engineering Task Force*) a proposé un nouveau standard, dénommé MIP (*Mobile IP*) [50][68]. Il est conçu afin de permettre à l'utilisateur de maintenir son adresse IP et par suite sa session d'accès aux services, tout au long de son déplacement d'un réseau IP à un autre. Selon le MIP, pour chaque réseau visité, l'utilisateur obtient une adresse IP temporaire, dénommée CoA (*Care of Address*), qui est différente de son adresse permanente, dénommée HoA (*Home Address*). Afin de gérer la mobilité de l'adresse IP associée au terminal en déplacement, trois acteurs interagissent : le MN (*Mobile Node*), le HA (*Home Agent*) qui est un routeur sur le réseau mère du terminal mobile, et le FA (*Foreign Agent*) qui est un routeur optionnel sur le réseau visité du terminal mobile. Ces trois acteurs permettent d'initier, de décider et d'exécuter le handover selon le processus suivant :

- Au début, le processus de gestion commence quand le MN reçoit un message, *Agent Advertisement*, de la part du FA qui correspond à la nouvelle localisation du terminal (réseau visité). Grâce à ce message, le FA attribue un nouveau CoA au MN. Ensuite, le MN initie le mécanisme de handover en envoyant une requête d'enregistrement (*Registration Request*) à son HA.
- Ainsi, le HA décide d'accepter ou de refuser l'enregistrement du MN au près du nouveau FA du réseau visité.
- Si le HA décide d'accepter l'enregistrement, il exécute le handover en enregistrant le *binding* (HoA,CoA) dans son *Binding Table* et en envoyant un *Registration*

Reply au MN. Par conséquent, si un CN (*Correspondent Node*) désire contacter le MN en utilisant l'adresse HoA, sa requête atteint le HA qui la redirige selon un tunnel IP vers l'adresse CoA du MN. Pour optimiser ce processus, un mécanisme de routage direct peut remplacer ce routage indirect. Pour ce faire, au lieu de rediriger la requête, le HA envoie au CN un message, *Binding Update*, dans lequel il indique le *binding* (HoA,CoA). Ainsi, le CN est capable de contacter directement le MN à l'aide d'un tunnel IP direct, sans devoir passer à chaque fois par le HA.

Enfin, nous constatons que le MIP gère seulement la mobilité du terminal et celle de l'utilisateur au niveau de la couche réseau cœur IP.

7.4 Défis des solutions de gestion de mobilité existantes

Après avoir analysé le contexte de gestion de mobilité, nous constatons que toutes les solutions existantes s'appuient sur un mécanisme de handover à trois rôles : initiateur, décideur et exécuteur. Ce mécanisme gère soit la mobilité de l'utilisateur, soit la mobilité du terminal, et s'applique soit au niveau des réseaux d'accès, soit au niveau du réseau cœur IP. Par conséquent, ces solutions réseaux gèrent la mobilité au niveau "*Media Delivery*". Autrement dit, ces solutions existantes proposent un mécanisme de handover qui maintient la continuité de transport du flux média et par suite la QoS, en optimisant le chemin pris par le flux média et en l'adoptant à tout évènement de mobilité spatiale (mobilité de l'utilisateur ou mobilité du terminal).

En effet, lors d'une dégradation d'une ressource réseau (congestion sur un routeur, diminution de la puissance du signal émis par une antenne, etc.), le niveau "*Media Delivery*" réagit dynamiquement pour maintenir la continuité d'accès et de transport et par suite la QoS. Cependant, la question qui se pose est la suivante : comment maintenir dynamiquement et sans couture la continuité de services lors d'une dégradation d'une ressource service ? De plus, au niveau du "*Media Delivery*", le mécanisme de handover intervient au niveau du réseau d'accès et au niveau du réseau cœur, de façon à relier le terminal de l'utilisateur à l'antenne et au routeur d'accès les plus adéquats pour le transport de ses données. Cependant, la question qui se pose aussi est : pourquoi ne pas avoir un handover au niveau service ?

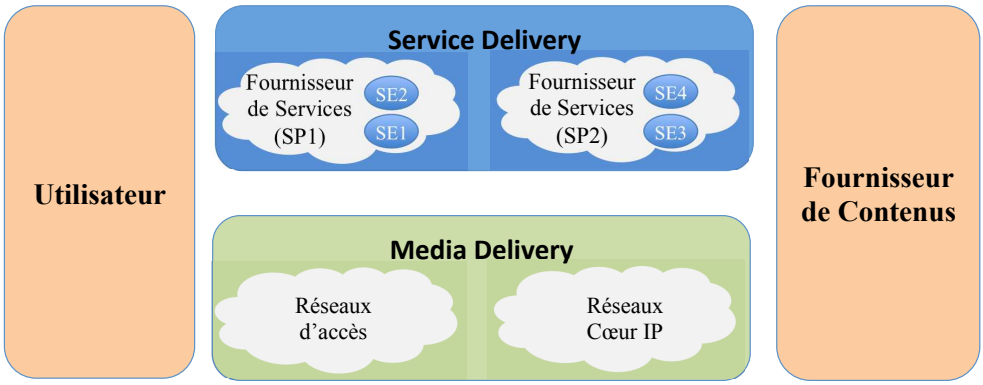

Fig. 7.1: Complémentarité du *Service Delivery* et du *Media Delivery*.

Suite à cette discussion, nous constatons qu'aucune des solutions existantes n'intervient au niveau de la couche service afin de traiter la continuité de services sans couture lors de la mobilité spatiale et temporelle de l'utilisateur, ou lors d'une dégradation de la QoS ou d'un mauvais fonctionnement d'une ressource service. Pour cela, nous proposons, comme le montre la Figure 7.1, un "*Service Delivery*" au dessus du "*Media Delivery*". Ce niveau "*Service Delivery*" permet de tenir compte de la couche service, et d'adapter d'une manière continue et transparente la session de services, par analogie au "*Media Delivery*" qui adapte les réseaux d'accès et réseau cœur, à tout évènement de mobilité et à tout changement dans le contexte ambiant de l'utilisateur.

Dans le chapitre suivant, nous proposons deux solutions qui sont considérées comme supports pour ce "*Service Delivery*". La première consiste à maintenir, d'une manière dynamique et transparente, la continuité de services lors d'une dégradation de la QoS ou d'un mauvais fonctionnement d'une ressource service. La deuxième est une extension du mécanisme de handover des couches réseaux à la couche service, afin d'adapter d'une manière continue et transparente la session de services de l'utilisateur à son nouveau contexte ambiant tout au long de sa mobilité spatiale et temporelle.

8 Proposition

Dans ce chapitre, nous levons un premier verrou qui consiste à maintenir, d'une manière transparente, la continuité de la session de services lors d'une dégradation de la QoS ou d'un mauvais fonctionnement d'un des services pré-provisionnés dans le VPSN de l'utilisateur. Pour ce faire, nous proposons une solution de gestion de mobilité qui est basée sur des communautés virtuelles (voir Section 8.1). Ensuite, nous levons un deuxième verrou qui consiste à prendre en considération le contexte ambiant de l'utilisateur, tout au long de sa mobilité spatiale et temporelle, tout en assurant une continuité sans couture de sa session de services. Pour ce faire, nous proposons une solution de gestion de mobilité qui est basée sur un handover sémantique (voir Section 8.2). Pour chacune de ces solutions, nous présentons les aspects architectural et fonctionnel, ainsi qu'une mise en œuvre qui relie ces deux aspects.

8.1 Les Communautés Virtuelles

8.1.1 Description du concept

Selon le journaliste politique Américain, *Norman Cousins*, la sagesse consiste à anticiper les conséquences. Cette stratégie s'applique non seulement dans la politique mais aussi dans tout domaine menacé par des erreurs. Dans notre approche *user-centric*, nous adoptons cette technique d'anticipation afin de prévenir toute discontinuité de la session de services de l'utilisateur. Puisque la cause de la coupure est soit le mauvais fonctionnement soit la dégradation de la QoS d'un des éléments de la session, nous basons notre anticipation sur le concept d'ubiquité. Ce dernier consiste à déployer d'une manière distribuer des éléments ayant la même fonctionnalité et une QoS équivalente.

Ainsi, afin de garantir la continuité de services tout en tenant compte des préférences fonctionnelles et non fonctionnelles (QoS) de l'utilisateur, nous proposons une solution de gestion de la mobilité, qui est pilotée par l'aspect d'ubiquité, et qui s'appuie sur une anticipation dynamique et transparente des dégradations des services pré-provisionnés dans le VPSN de l'utilisateur. En effet, cette solution consiste à regrouper des services

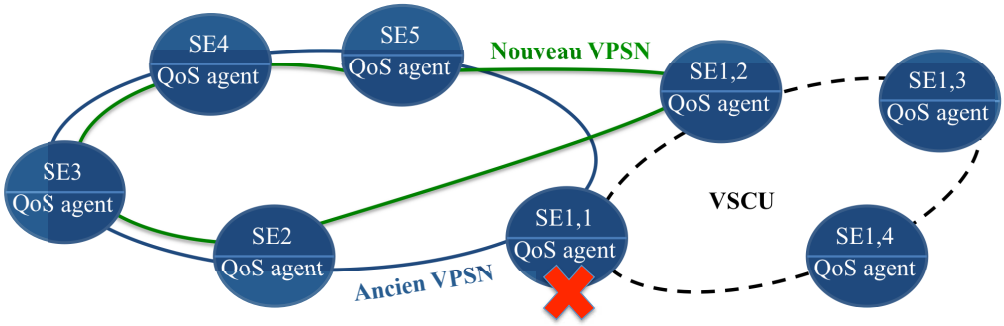

Fig. 8.1: Gestion de la mobilité par les Communautés Virtuelles.

ubiquitaires dans des communautés virtuelles, dénommées VSCUs (*Ubiquity-based Virtual Service Communities*). Par conséquent, quand un service sélectionné ne peut plus continuer à vérifier les préférences d'un certain utilisateur, il sera remplacé dans le VPSN de ce dernier par un autre service qui lui est ubiquitaire et qui appartient à sa VSCU. Ainsi, nous maintenons par anticipation et d'une manière dynamique et transparente la continuité de la session de services de l'utilisateur, tout en respectant ses préférences au niveau fonctionnel et au niveau de la QoS demandée.

Ce principe de gestion de mobilité par les communautés virtuelles n'est pas limité à la couche service. Il est appliqué aussi sur la couche réseau et sur la couche des équipements. Ainsi, nous avons conçu, par analogie aux VSCUs, des VCCUs et des VECUs qui créent respectivement des communautés virtuelles au niveau des connectivités réseaux et au niveau des équipements. Cependant, dans le cadre de cette thèse, nous nous intéressons plus à la partie "*Service Delivery*". Pour cela, dans la suite de cette partie, nous nous focalisons sur la couche service et nous traitons seulement le cas des VSCUs.

Cette gestion de mobilité à l'aide des communautés virtuelles de niveau service est présentée dans la Figure 8.1. Nous considérons le cas d'un utilisateur ayant un VPSN pré-provisionnant les services suivants : {*SE1,1* ; *SE2* ; *SE3* ; *SE4* et *SE5*}. Nous supposons que le service *SE1,1* subit un mauvais fonctionnement ou une dégradation de sa QoS. Il n'arrive pas à respecter son contrat de SLA déjà négocié avec l'utilisateur. Afin d'empêcher toute discontinuité de la session de services, nous anticipons en remplaçant le service dégradé par un autre service ubiquitaire, par exemple *SE1,2*, appartenant à sa communauté VSCU créée en avance et contenant les services suivant : {*SE1,1* ; *SE1,2* ; *SE1,3* ; *SE1,4*}. Cependant, la question qui se pose est la suivante : comment effectuer ce remplacement pour gérer la mobilité de la façon la plus rapide, la plus dynamique et la plus transparente ?

Pour répondre à cette question, nous adoptons une approche évènementielle qui s'appuie sur trois acteurs : un initiateur, un décideur et un exécuteur. Cette approche évènementielle est renforcée par une autre originalité qui consiste à avoir des éléments de service autogérés. En effet, l'autogestion de chaque service est assurée par un agent de QoS qui est intégré au niveau du plan de gestion de ce service. Cet agent compare la QoS courante à la valeur seuil à ne pas dépasser. Si elle est supérieure, il envoie un *IN-Contract* qui montre que le comportement courant du service est toujours conforme

à celui négocié dans le contrat de SLA, sinon il envoie un *OUT-Contract*. Dans ce deuxième cas, c'est l'agent de QoS du service dégradé qui détecte l'évènement de non respect du contrat signé et qui joue ainsi le rôle d'initiateur du processus de gestion par VSCUs. Ensuite, c'est lui qui va jouer le rôle de décideur en sélectionnant, parmi les membres du VSCU, un remplaçant ubiquitaire. Une fois la décision prise, le service *VPSN Maintenance* intervient pour jouer le rôle d'exécuteur en adaptant le VPSN en question. Cependant, la problématique qui se pose est de savoir comment le décideur va choisir le remplaçant ubiquitaire. Pour lever ce défi, deux algorithmes semblent possibles :

- Le premier algorithme consiste à ce que le service dégradé envoie l'*OUT-Contract* en *multicast*, c.à.d. il l'envoie vers tous les services ubiquitaires qui se trouvent dans sa VSCU. Ensuite, il reçoit la réponse de chacun d'entre eux, et il choisit comme remplaçant le service ubiquitaire qui a répondu le premier.
- Le deuxième algorithme consiste à ce que le service dégradé envoie l'*OUT-Contract* au premier service ubiquitaire de sa table de QoS, faisant partie de sa VSCU. Cette table classifie les services ubiquitaires du plus proche au plus loin d'un point de vue réseau, en se basant sur une QoS_{totale} qui est une agrégation de la QoS du service contenant cette table avec la QoS du service ubiquitaire ainsi que la QoS du lien qui les relie.

Il faut noter qu'un service peut être pré-provisionné dans plusieurs VPSN correspondant à différents utilisateurs. Par conséquent, un service qui ne peut plus vérifier le contrat de SLA signé avec un utilisateur donné, peut toujours rester conforme à d'autres contrats signés avec d'autres utilisateurs. Ainsi, le processus de maintenance ne sera exécuté que pour le VPSN vis-à-vis duquel le service est dégradé.

Après avoir décrit le concept de communautés virtuelles basées sur l'ubiquité, de nouvelles questions se posent : n'y a-t-il pas des besoins autres que l'ubiquité qui puissent influencer la création des communautés virtuelles ? Comment structurer ces communautés virtuelles au niveau architectural, et comment les créer et les gérer au niveau fonctionnel ?

8.1.2 Aspect architectural des VSCUs

8.1.2.1 CoI : VSC-U&P, VSC-U&L et VSCU-L&P

L'approche d'anticipation que nous adoptons pour gérer la continuité de services s'appuie sur le principe des communautés virtuelles. Lors de cette gestion de mobilité, nous avons besoin de respecter les préférences fonctionnelles de l'utilisateur ainsi que la QoS demandée. Pour cela, les communautés virtuelles conçues ont intérêt à regrouper des services ubiquitaires, d'où la notion des VSCUs. Par conséquent, nous considérons ces VSCUs comme des communautés d'intérêt, notée CoI (*Community of Interest*), qui regroupent des éléments de services partageant un intérêt commun qui est celui de l'ubiquité afin de respecter les préférences de l'utilisateur lors de la gestion de la mobilité. Cependant, l'ubiquité n'est pas le seul intérêt selon lequel nous pouvons regrouper les éléments de services. En effet, différents types de CoI peuvent être conçus

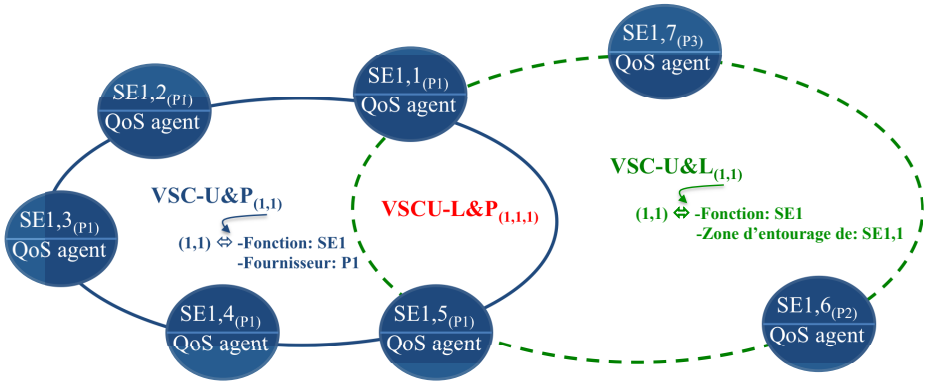

Fig. 8.2: Communautés d'intérêts : VSC-U&P, VSC-U&L et VSCU-L&P.

pour répondre à différents besoins, tout en tenant compte de l'objectif principal qui est de maintenir la continuité de services tout au long de la session de l'utilisateur.

Comme le montre la Figure 8.2, nous proposons deux types de CoI :

- VSC-U&P (*Ubiquity and Provider based Virtual Service Community*) : elle reflète le besoin du fournisseur à maintenir la continuité de services tout en remplaçant chacun de ses services dégradés par un autre service ubiquitaire qui lui appartient. En effet, cette approche lui permet de gérer la mobilité de ses services et de maintenir les préférences de ses abonnés, sans être obligé de passer par un autre fournisseur et de lui payer l'utilisation d'un de ses services ubiquitaires. Il faut noter que cette approche est favorisée par notre hypothèse qui considère que chaque fournisseur doit distribuer, à grande échelle et selon sa stratégie dans le marché des services, ses services ubiquitaires dans différentes zones géographiques pour répondre aux demandes du plus grand nombre d'utilisateurs. Lorsque le fournisseur trouve selon son étude du marché et selon sa stratégie économique que les services ubiquitaires déployés ne suffisent pas, il signe des contrats de *peering* avec d'autres fournisseurs qui offrent des services ubiquitaires. Afin de répondre à ce besoin de la part des fournisseurs de services, nous concevons les VSC-U&Ps. En effet, chaque VSC-U&P regroupe des éléments de service ayant l'ubiquité et l'identité du fournisseur comme intérêts communs. Par conséquent, ces éléments de service sont ubiquitaires et appartiennent à un même fournisseur de services. D'après la Figure 8.2, nous montrons un fournisseur *P1* qui regroupe une liste d'éléments de services ubiquitaires $\{SE1,i_{(P1)}$; *Service Elements* avec i allant de 1 à 5$\}$ dans une communauté virtuelle VSC-U&P$_{1,1}$ où le couple (1,1) représente respectivement le type de service (*SE1*) et le type de fournisseur (*Provider* : *P1*).
- VSC-U&L (*Ubiquity and Location based Virtual Service Community*) : elle reflète le besoin de l'utilisateur à avoir une session de services continue qui remplace tout service dégradé du VPSN par un autre service ubiquitaire qui appartient à l'entourage géographique du service dégradé. En effet, grâce à cette approche, l'utilisateur sera capable de conserver sa QoS de bout-en-bout, puisque le service ubiquitaire remplaçant est à proximité du service remplacé, ce qui implique

que tous les deux s'appuient normalement sur des contextes réseaux équivalents et par suite ils garantissent des temps de latence aussi équivalents. Afin de répondre à ce besoin de la part des utilisateurs, nous concevons les VSC-U&Ls. En effet, chaque VSC-U&L regroupe des éléments de service ayant l'ubiquité et la localisation géographique des services comme intérêts communs. Par conséquent, ces éléments de service sont ubiquitaires et appartiennent à une même zone géographique. Cependant, ils peuvent appartenir à différents fournisseurs de services. D'après la Figure 8.2, nous montrons aussi une liste de services ubiquitaires $\{SE1{,}1_{(P1)}, SE1{,}5_{(P1)}, SE1{,}6_{(P2)}, SE1{,}7_{(P3)}\}$ qui sont regroupés dans une communauté virtuelle VSC-U&$L_{1,1}$ où le couple (1,1) représente dans ce cas respectivement le type de service (*SE1*) et la localisation géographique du service $SE1{,}1_{(P1)}$ autour duquel la communauté est créée.

Après avoir décrit ces deux types de CoI, nous les combinons pour obtenir un troisième type de CoI, noté VSCU-L&P (*Ubiquity, Location and Provider based Virtual Service Community*). Cette VSCU-L&P combine des éléments de services ubiquitaire, appartenant au même fournisseur et se trouvant dans une même zone géographique. Par conséquent, grâce à ce troisième type de CoI, la continuité de services est maintenue tout en tenant compte des préférences de l'utilisateur et des différents besoins qui régissent le point de vue du fournisseur et celui de l'utilisateur. D'après la Figure 8.2, nous montrons que la combinaison de la VSC-U&$P_{1,1}$ et de la VSC-U&$L_{1,1}$ donne une nouvelle communauté virtuelle notée VSCU-L&$P_{1,1,1}$ où le triplet (1,1,1) représente respectivement le type de service (*SE1*), la localisation géographique du service $SE1{,}1_{(P1)}$ autour duquel la communauté est créée, et le type de fournisseur (*P1*).

Dans la suite, nous montrons comment ces différents types de communautés virtuelles vont être créés et gérés de façon à contenir toujours les services ubiquitaires qui vérifient les intérêts voulus. Cependant, pour pouvoir lancer le processus de création et de gestion des différentes CoI, nous avons besoin d'identifier le responsable pour chacune d'elles. En effet, au niveau de la VSC-U&P et de la VSCU-L&P, le responsable qui va créer et gérer ces communautés est le fournisseur qui dans ce cas est commun pour tous les services ubiquitaires sélectionnés. Or, ce n'est pas pareil pour la VSC-U&L dans laquelle les services ubiquitaires peuvent appartenir à différents fournisseurs. Par conséquent, la problématique qui se pose est de se savoir qui est responsable de la création et de la gestion d'une VSC-U&L multi-fournisseurs.

8.1.2.2 Inner VSC-U&L et Outer VSC-U&L

Dans le cas des communautés virtuelles VSC-U&Ls, les services ubiquitaires sélectionnés peuvent appartenir non seulement à des fournisseurs différents, mais aussi à des plates-formes de services (des NGN/NGS Middleware) différentes au sein d'un même fournisseur. Cette approche est de plus en plus favorisée par la distribution à grande échelle qu'adoptent les fournisseurs pour déployer leurs services ubiquitaires, et par les liens de *peering* qui sont créés entre eux et qui permettent plus de collaboration distribuée. Afin d'accompagner cette distribution de services ubiquitaires et la création de VSC-U&Ls trans-organisationnelles, nous avons besoin d'une structure qui supporte

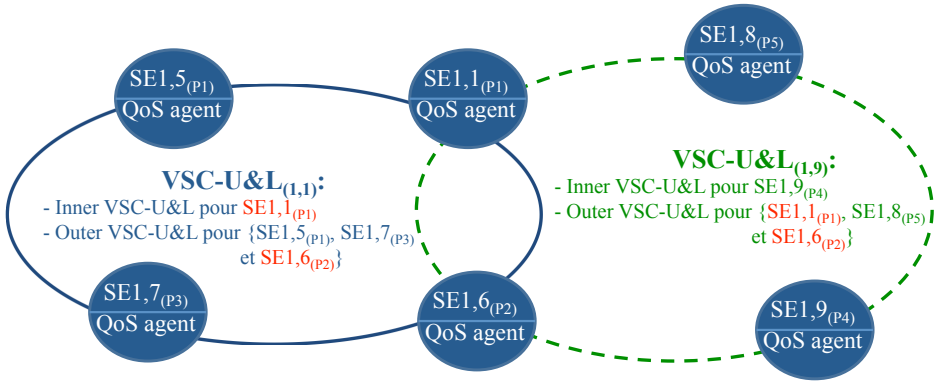

Fig. 8.3: Inner VSC-U&L contre Outer VSC-U&L.

une gestion distribuée de ces VSC-U&Ls.

Pour dépasser ce défi de création et de gestion distribuées des VSC-U&Ls trans-organisationnelles, nous proposons deux types de VSC-U&L :

- Inner VSC-U&L : elle est *Inner* vis-à-vis d'un service donné et vis-à-vis de sa plate-forme, quand elle est associée au déploiement de ce service dans cette plate-forme. En effet, lors du déploiement de ce service, sa plate-forme crée son Inner VSC-U&L à partir de services ubiquitaires multi-fournisseurs appartenant à sa zone d'entourage (*Surrounding Zone*). Cette dernière correspond à un cercle de rayon R_s (*Surrounding Radius*) et ayant comme centre l'adresse géographique (Longitude,Latitude) du service en question. Ensuite, c'est cette même plate-forme qui sera considérée comme le responsable de gestion de cette Inner VSC-U&L créée. Il faut noter qu'un service appartient à une seule Inner VSC-U&L qui est considérée comme interne (*Inner*) pour sa plate-forme. En effet, cette dernière ne gère que ses propres Inner VSC-U&Ls, c.à.d. celles qui sont créées en interne suite au déploiement d'un de ses propres services.
- Outer VSC-U&L : elle est *Outer* vis-à-vis de certains services et de leurs plates-formes, quand elle n'est pas associée au déploiement de ces services. En effet, lors du déploiement d'un service donné, sa plate-forme crée son Inner VSC-U&L à partir de services ubiquitaires multi-fournisseurs appartenant à sa zone d'entourage. Par conséquent, cette VSC-U&L qui est *Inner* pour le service déployé et pour sa plate-forme, est aussi *Outer* pour les autres membres de cette même communauté et pour leurs plates-formes. Donc, la notion d'*Inner* et d'*Outer* varie selon le point de vision des services et des plates-formes. Il faut noter qu'un service peut appartenir à plusieurs Outer VSC-U&Ls qui sont considérées comme externes (*Outer*) pour sa plate-forme. En effet, cette dernière n'est pas responsable de la gestion de ces Outer VSC-U&Ls. Ces communautés sont gérées seulement par les plates-formes qui les ont créées.

Afin de mieux différencier ces deux types de VSC-U&L, nous présentons un exemple dans la Figure 8.3. Nous considérons un service $SE1,1_{(P1)}$ qui appartient à une des plates-formes du fournisseur *P1*. Lors de son déploiement, sa plate-forme crée une communauté virtuelle $\text{VSC-U\&L}_{1,1}$ qui contient, en plus de $SE1,1_{(P1)}$, la liste des services

$\{SE1{,}5_{(P1)}, SE1{,}6_{(P2)}, SE1{,}7_{(P3)}\}$ qui sont ubiquitaires, multi-fournisseurs et qui appartiennent à l'entourage géographique de $SE1{,}1_{(P1)}$. Puisque c'est $SE1{,}1_{(P1)}$ et sa plate-forme qui ont causé la création de VSC-U&$L_{1,1}$, cette dernière est considérée, d'une part comme une Inner VSC-U&L pour $SE1{,}1_{(P1)}$ et pour sa plate-forme, et d'autre part comme une Outer VSC-U&L pour chacun des autres services ubiquitaires et pour leurs plates-formes. Par conséquent, seule la plate-forme de $SE1{,}1_{(P1)}$ sera responsable de la gestion de VSC-U&$L_{1,1}$. En outre, dans ce même exemple, nous considérons le déploiement d'un service $SE1{,}9_{(P4)}$ qui appartient à une des plates-formes du fournisseur *P4*. Lors de son déploiement, sa plate-forme crée une communauté virtuelle VSC-U&$L_{1,9}$ qui contient, en plus de $SE1{,}9_{(P4)}$, la liste des services $\{SE1{,}1_{(P1)}, SE1{,}6_{(P2)}, SE1{,}8_{(P5)}\}$ qui sont ubiquitaires, multi-fournisseurs et qui appartiennent à l'entourage géographique de $SE1{,}9_{(P4)}$. Par conséquent, VSC-U&$L_{1,9}$ est considérée comme Inner VSC-U&L pour $SE1{,}9_{(P4)}$ et comme Outer VSC-U&L pour les autres. Ainsi, nous avons montré qu'un service, comme $SE1{,}1_{(P1)}$, peut appartenir en même temps à une Inner VSC-U&L et à une Outer VSC-U&L. Sa plate-forme sera seulement responsable de l'Inner VSC-U&L. De plus, nous avons montré qu'un service, comme $SE1{,}6_{(P2)}$, peut appartenir en même temps à plusieurs Outer VSC-U&Ls (selon l'exemple, il appartient à VSC-U&$L_{1,1}$ et VSC-U&$L_{1,9}$).

Par conséquent, cette approche d'Inner VSC-U&L et d'Outer VSC-U&L montre bien la structuration architecturale distribuée que nous proposons pour la création et la maintenance des communautés virtuelles. Dans la suite, nous proposons la liste de fonctionnalités qui vont permettre de gérer la continuité de services en créant et maintenant les différents types de communautés virtuelles.

8.1.3 Aspect fonctionnel des VSCUs

Afin de maintenir la continuité de services tout en tenant compte des préférences de l'utilisateur, nous avons proposé une solution de gestion basée sur différents types de VSCUs. Cependant, la question qui se pose est : comment créer des VSCUs, comment attacher des nouveaux services déployés à des VSCUs existantes, et comment maintenir ces VSCUs ? Pour répondre à cette question, nous proposons les services de base suivant : le *VSCU Attachment*, le *VSCU Creation*, l'*Internal VSCU Maintenance* et l'*External VSCU Maintenance*. Ces services de gestion permettent respectivement d'attacher un service déployé à une VSCU existante, de créer une nouvelle VSCU et de maintenir une VSCU créée. En outre, ces services vérifient une approche évènementielle qui consiste à agir lorsqu'ils reçoivent les notifications correspondant aux évènements auxquels ils sont abonnés.

Il faut noter que les VSCUs que nous traitons dans cette partie sont les VSC-U&Ps et les VSC-U&Ls. De plus, suite à notre approche de gestion distribuée, nous allons décrire comment une certaine plate-forme va traiter les VSC-U&Ls dont elle est responsable. Par conséquent, les VSC-U&Ls traitées sont plutôt les Inner VSC-U&Ls pour cette plate-forme de gestion.

8.1.3.1 Attachement aux VSCUs

Lors du déploiement d'un nouveau service dans une des plates-formes d'un fournisseur donné, le service *VSCU Attachment* de cette plate-forme essaye d'attacher le service déployé à une des VSC-U&Ps relatives à ce fournisseur et à une des Inner VSC-U&Ls relatives à cette plate-forme. En effet, l'intérêt de cet attachement est d'éviter, si possible, la création de nouvelles VSCUs pour chaque déploiement d'un nouveau service.

Pour pouvoir lancer ce processus d'attachement, l'évènement de déploiement d'un nouveau service dans une plate-forme doit être détecté. Pour cette raison, nous associons le déploiement du service à la création d'un *Service Profile* dans l'*Infoware* relatif à cette plate-forme. Ce *Service Profile* contient les champs suivants {*Service ID, Functionality, QoS criteria, Provider ID, SIP URI, Longitude, latitude, VSC-U&P, Inner VSC-U&L, Outer VSC-U&Ls*}, ainsi que le rayon R_c qui délimite la zone de couverture de ce service et qui sera utilisé dans la partie de handover sémantique. Pour l'instant, les champs {*VSC-U&P, Inner VSC-U&L, Outer VSC-U&Ls*} restent vides. Par conséquent, suite à la création de ce *Service Profile*, l'*Infoware* détecte l'évènement de déploiement du nouveau service dans la plate-forme, et notifie l'*Events Manager* relatif à cette dernière en envoyant une *Service Deployment Notification* contenant le *Service ID*. Ainsi, l'*Events Manager* associe cet évènement à l'action de notifier le service *VSCU Attachment* abonné à ce type d'évènement.

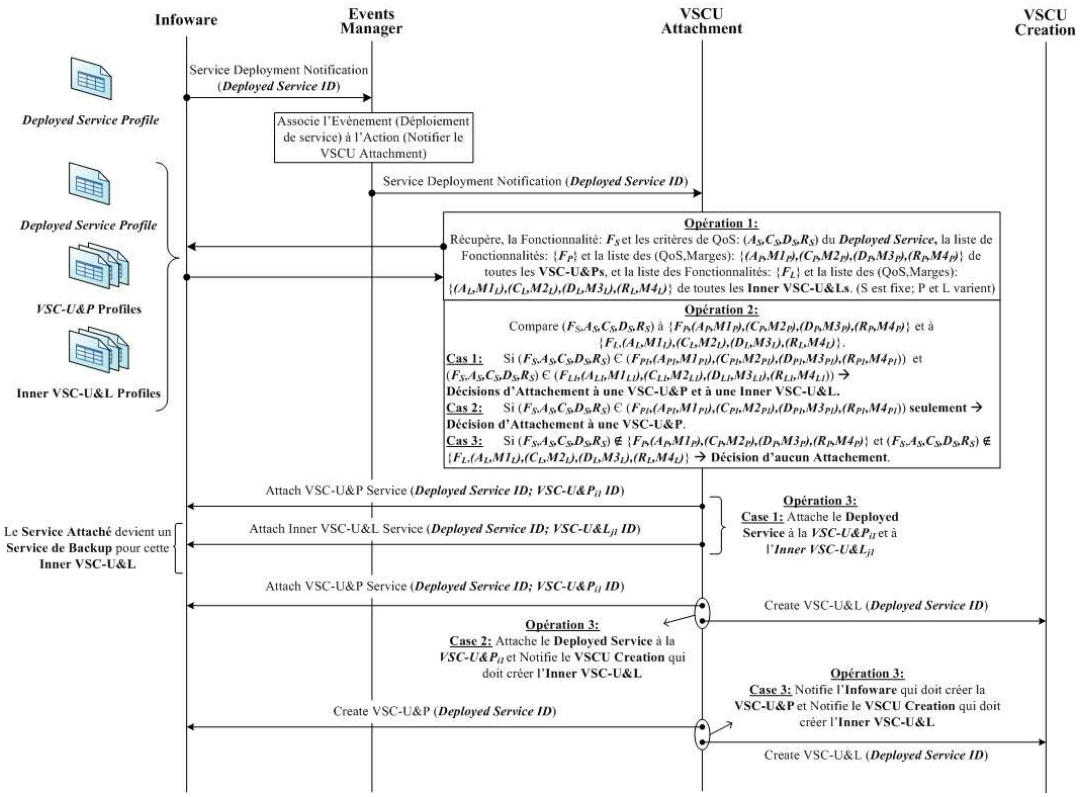

Fig. 8.4: Diagramme de séquence pour l'attachement aux VSCUs.

Comme le montre la Figure 8.4, le processus fonctionnel du service *VSCU Attachment* correspond à la séquence suivante d'opérations :

- Opération 1 : Le *VSCU Attachment* envoie une requête à l'*Infoware* pour récupérer, d'une part la liste {*Functionality, QoS criteria*} relative au service déployé tout en s'appuyant sur son *Service Profile*, et d'autre part les listes {*Functionality, QoS criteria, QoS margins*} relatives aux différentes VSC-U&Ps et Inner VSC-U&Ls tout en s'appuyant sur leurs profils. Il faut noter que dans les profils des communautés virtuelles, nous associons aux quatre critères de QoS quatre marges de QoS, car ces communautés combinent des services ayant des valeurs de QoS qui ne sont pas forcément identiques, mais plutôt équivalentes.
- Opération 2 : Le *VSCU Attachment* compare la fonctionnalité et les valeurs de QoS relatives au service déployé à celles des VSC-U&Ps et des Inner VSC-U&Ls. Ainsi, trois cas sont possibles. Le premier cas se produit quand les caractéristiques du service déployé vérifient celles d'une Inner VSC-U&L et d'une VSC-U&P. Le deuxième cas se produit quand les caractéristiques du service déployé ne vérifient celles d'aucune Inner VSC-U&L, mais elles vérifient celles d'une VSC-U&P. Le troisième cas se produit quand les caractéristiques du service déployé ne vérifient celles d'aucune Inner VSC-U&L ni celles d'aucune VSC-U&P. Il faut noter qu'un quatrième cas, qui consiste à avoir un service déployé dont la fonctionnalité et les valeurs de QoS vérifient celles d'une Inner VSC-U&L mais ne vérifient celles d'aucune VSC-U&P, ne se produira jamais. En effet, le fait que les caractéristiques du service déployé vérifient celles d'une Inner VSC-U&L implique qu'il y a déjà dans cette même plate-forme un service ubiquitaire qui a été déjà déployé et qui a causé la création de cette Inner VSC-U&L et qui ensuite a sûrement causé la création d'une certaine VSC-U&P.
- Opération 3 : Pour le premier cas, le *VSCU Attachment* envoie les notifications suivantes {*Attach Inner VSCU&L Service, Attach VSC-U&P Service*} à l'*Infoware* tout en précisant les identifiants du service déployé, de l'Inner VSC-U&L et de la VSC-U&P. Ainsi, l'*Infoware* effectue l'attachement du service déployé à l'Inner VSC-U&L et à la VSC-U&P. Pour le deuxième cas, le *VSCU Attachment* envoie une première notification, *Attach VSC-U&P Service*, à l'*Infoware* tout en précisant les identifiants du service déployé et de la VSC-U&P. Ainsi, l'*Infoware* effectue l'attachement du service déployé à la VSC-U&P. De plus, il envoie une deuxième notification, *Create VSC-U&L*, au service *VSCU Creation* de la même plate-forme tout en précisant l'identifiant du service déployé. Ainsi, le service *VSCU Creation* crée selon son propre processus fonctionnel l'Inner VSC-U&L correspondant à ce nouveau service déployé. Enfin, pour le troisième cas, le *VSCU Attachment* envoie une notification, *Create VSC-U&L*, au service *VSCU Creation* comme dans le deuxième cas, afin que ce dernier crée selon son processus l'Inner VSC-U&L correspondante. De plus, le *VSCU Attachment* envoie une notification, *Create VSC-U&P*, directement à l'*Infoware* afin que ce dernier crée la VSC-U&P relative au service déployé. Il faut noter que pour la création d'une VSC-U&P, nous n'avons pas besoin de passer par le service *VSCU Creation*, car l'*Infoware* possède toutes les informations nécessaires pour créer par inférence

cette VSC-U&P.

Dans le cas où l'*Infoware* crée la VSC-U&P relative au service déployé, il crée un *VSC-U&P Profile* dans lequel il met les informations suivantes {*VSC-U&P ID, Functionality, QoS criteria, QoS margins, Provider ID*}. Ces informations sont prises par inférence d'après le *Service Profile* du service déployé. Ce *Service Profile* change aussi par inférence en ajoutant l'identifiant de la VSC-U&P. Il faut noter que la VSC-U&P dans ce cas contient seulement un seul service. Puisque la VSC-U&P créée concerne un fournisseur et non pas une plate-forme bien déterminée, l'*Infoware* envoie des notifications, *VSC-U&P Creation*, à tous les fragments d'*Infoware* relatifs aux différentes plates-formes de ce fournisseur. Dans ces notifications, l'*Infoware* précise toutes les informations qui concernent la VSC-U&P créée (le *VSC-U&P Profile* ainsi que l'identifiant du service déployé, son adresse logique et sa zone de couverture qui sera utilisée par les autres plates-formes pour le handover sémantique).

Dans le cas où l'*Infoware* exécute l'attachement du service déployé à la VSC-U&P, il ajoute dans le *Service Profile* l'identifiant de la VSC-U&P. Ensuite, il envoie des notifications de mise à jour, *VSC-U&P Service Attachment*, vers tous les fragments d'*Infoware* relatifs aux différentes plates-formes de ce fournisseur. Cette mise à jour contient l'identifiant de la VSC-U&P adaptée ainsi que l'identifiant du service attaché, son adresse logique et sa zone de couverture.

Enfin, dans le cas où l'*Infoware* exécute l'attachement du service déployé à l'Inner VSC-U&L, il ajoute le *Service ID* à l'*Inner VSC-U&L Profile* et l'*Inner VSC-U&L ID* au *Service Profile*. Ainsi, le service déployé considère la communauté virtuelle à laquelle il vient de s'attacher comme son Inner VSC-U&L. Or, cette dernière a été déjà créée suite au déploiement d'un autre service dans cette même plate-forme. Par conséquent, nous nous trouvons dans une situation où nous avons deux services d'une même plate-forme partageant la même Inner VSC-U&L. Nous différencions ces deux services en considérant celui qui était à l'origine de cette Inner VSC-U&L, comme un responsable, quand au deuxième service, nous le considérons comme un *Backup*. Ainsi, les autres services qui appartiennent à cette communauté et qui la considère comme Outer VSC-U&L n'auront pas conscience de l'existence de ce *Backup*. Par contre, ils ont des informations sur le responsable de cette communauté et ils le contactent en cas de dégradation.

8.1.3.2 Création des VSCUs

Lors du déploiement d'un nouveau service $SE_{i1,j1,Px}$ dans une des plates-formes d'un fournisseur P_x, si le service *VSCU Attachment* n'arrive pas à attacher $SE_{i1,j1,Px}$ à une Inner VSC-U&L, il envoie une notification, *Create VSC-U&L*, au service *VSCU Creation* appartenant à cette même plate-forme. Ainsi, le *VSCU Creation* crée une Inner VSC-U&L relative au service $SE_{i1,j1,Px}$ déployé. Ce dernier sera considéré comme le responsable de cette communauté.

Comme le montre la Figure 8.5, le processus fonctionnel du service *VSCU Creation* correspond à la séquence suivante d'opérations :

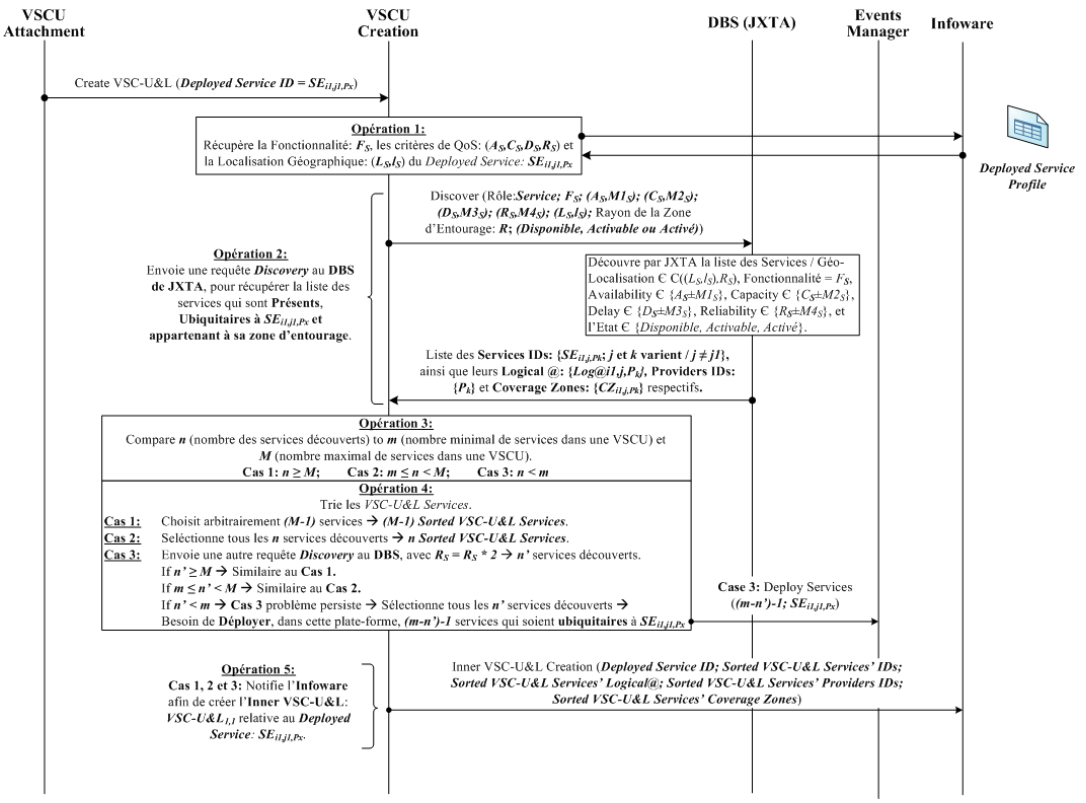

Fig. 8.5: Diagramme de séquence pour la création des VSCUs.

- Opération 1 : Le *VSCU Creation* envoie une requête à l'*Infoware* pour récupérer la liste {*Functionality, QoS criteria, Longitude, latitude*} relative au service $SE_{i1,j1,Px}$ déployé tout en s'appuyant sur son *Service Profile*.
- Opération 2 : Le *VSCU Creation* envoie une requête au service découverte DBS de JXTA. Ce dernier découvre la liste des services $\{SE_{i1,j,Pk}$: j et k varient, avec $j \neq j1\}$ qui sont présents (disponibles, activables ou activés), ubiquitaires au service déployé, et qui appartiennent à sa zone d'entourage géographique. Pour ce faire, le service DBS reçoit de la part du service *VSCU Creation* les entrée suivantes : le rôle = "service" et la liste des critères de découverte = {la fonctionnalité du service déployé, ses quatre critères de QoS avec les quatre marges relatives, sa longitude, sa latitude, le rayon R_s de la zone d'entourage, ainsi que l'état disponible, activable ou activé}. Ensuite, le DBS envoie comme sortie la liste des identifiants des services découverts, leurs adresses logiques (SIP URIs) pour qu'ils puissent être contactés, leurs *Provider IDs* puisqu'ils sont multi-fournisseurs, ainsi que leurs zones de couverture qui seront utilisées dans la suite pour le handover sémantique.
- Opération 3 : Quand le *VSCU Creation* reçoit la réponse du DBS, il compare le nombre "n" de services découverts à une valeur "m" minimale et une valeur "M" maximale. En effet, afin de ne pas encombrer la base de connaissances par des VSC-U&Ls avec un grand nombre de services et afin de ne pas avoir aussi des

VSC-U&Ls à faible nombre de services ubiquitaires, chaque fournisseur délimite, selon ses études du marché et ses heuristiques, le nombre acceptable de services ubiquitaires dans une VSC-U&L entre un nombre minimal "m" et un nombre maximal "M". Suite à cette comparaison, trois cas sont possibles : soit "n" est supérieur à "M", soit "n" est entre "m" et "M", soit "n" est inférieur à "m".

- Opération 4 : Dans le premier cas, le *VSCU Creation* choisit arbitrairement "M-1" services. Dans le deuxième cas, il prend tous les "n" services découverts. Quand au troisième cas, il relance la découverte avec le service DBS mais cette fois en multipliant par deux le rayon R_s de la zone d'entourage. Si après la deuxième découverte le nombre "n" est devenu supérieur à "M" ou bien entre "m" et "M", nous passons ainsi à une situation conforme au premier cas ou au deuxième cas. Cependant, si après cette deuxième découverte, le problème persiste et nous avons toujours un nombre "n" inférieur à "m", nous ne lançons pas une troisième fois la découverte DBS en augmentant le rayon, car dans ce cas nous perdons l'intérêt de proximité géographique entre les services ubiquitaires sélectionnés par la VSC-U&L. Pour dépasser ce problème, nous proposons de déployer dans cette même plate-forme un nombre "(m-n)-1" services ubiquitaires au service $SE_{i1,j1,Px}$. Pour ce faire, le *VSCU Creation* envoie une notification, *Deploy Services*, à l'*Events Manager* tout en précisant le nombre "(m-n)-1" de services à déployer ainsi que l'identifiant du service $SE_{i1,j1,Px}$ auquel ils doivent être ubiquitaires.
- Opération 5 : Enfin, pour les trois cas possibles, le *VSCU Creation* envoie une notification, *Inner VSC-U&L Creation*, directement à l'*Infoware* afin que ce dernier crée l'Inner VSC-U&L relative au service $SE_{i1,j1,Px}$ déployé. Dans cette notification, le *VSCU Creation* précise l'identifiant du service $SE_{i1,j1,Px}$ déployé, la liste des identifiants des services ubiquitaires $\{SE_{i1,j,Pk} : j$ et k varient, avec $j \neq j1\}$, leurs adresses logiques, leurs *Provider IDs* ainsi que leurs zones de couverture.

Ainsi, l'*Infoware* crée l'Inner VSC-U&L relative au service $SE_{i1,j1,Px}$ déployé, en créant un *VSC-U&L Profile* dans lequel il met les informations suivantes {*VSC-U&L ID, Functionality, QoS criteria, QoS margins, Longitude, Latitude,* R_s*, l'identifiant de* $SE_{i1,j1,Px}$ *considéré comme responsable*}. En effet, la plupart de ces informations sont prises par inférence d'après le *Service Profile* du service $SE_{i1,j1,Px}$ déployé. Ce *Service Profile* change aussi par inférence en ajoutant, dans le champ Inner VSC-U&L, l'identifiant de la VSC-U&L créée.

Enfin, nous obtenons une Inner VSC-U&L qui est relative à un service $SE_{i1,j1,Px}$ donné, et qui contient une liste de services $\{SE_{i1,j,Pk} : j$ et k varient, avec $j \neq j1\}$ multi-fournisseurs, auxquels sont associées des listes d'adresses logiques, de *Provider IDs* ainsi que des zones de couverture. Par conséquent, l'*Infoware* notifie les autres fragments d'*Infoware* relatifs aux plates-formes de ces services multi-fournisseurs, en envoyant des notifications, *Outer VSC-U&L Creation*, dans lesquelles il précise l'identifiant de la VSC-U&L créée, l'identifiant du service $SE_{i1,j1,Px}$ responsable ainsi que son adresse logique. Suite à cette notification, chaque fragment associe la VSC-U&L créée à chacun de ces services. Ces derniers la considère comme étant une Outer VSC-U&L dont le responsable est un service $SE_{i1,j1,Px}$ externe.

D'autre part, il faut aussi noter que dans le cas où le service *VSCU Creation* envoie

la notification, *Deploy Services*, à l'*Events Manager*, ce dernier notifie à son tour un certain service de gestion qui est responsable de déployer les "(m-n)-1" services. Ainsi, pour chacun des "(m-n)-1" services ubiquitaires créés, le service *VSCU Attachment* va être sollicité. Ce dernier va normalement attacher chacun de ces services à l'Inner VSC-U&L qui a été déjà créée par le *VSCU Creation* et qui a été déjà associée au service $SE_{i1,j1,Px}$. Par conséquent, en attachant ces "(m-n)-1" services à l'Inner VSC-U&L, nous obtenons "m" services dans cette communauté au lieu des "n" services (où "n" était inférieur à "m").

8.1.3.3 Maintenance des VSCUs

Après avoir créé ses propres Inner VSC-U&Ls et avoir participé à la création de la VSC-U&P relative à son fournisseur, chaque plate-forme doit maintenir ces différents types de communautés. Pour ce faire, des services de maintenance des VSCUs sont conçus dans chacune de ces plates-formes. Ces services de gestion s'appuient sur les *OUT-Contracts* envoyés par les agents de QoS des services autogérés. Cependant, il faut différencier la maintenance des communautés de la maintenance du VPSN de l'utilisateur qui est plus prioritaire. En effet, la maintenance du VPSN doit être lancée dès le premier *OUT-Contract* envoyé par un des services pré-provisionnés dans ce VPSN. Or, ce service peut faire simultanément partie d'autres VPSNs qui le considèrent toujours conforme aux contrats de ses utilisateurs. Pour cela, un seul *OUT-Contract* envoyé par un service donné n'est pas suffisant pour considérer ce service comme incapable de répondre à des nouvelles demandes. Ainsi, nous le gardons dans les communautés virtuelles qui l'ont sélectionné, et nous ne lançons le processus de maintenance de ces communautés que lorsque ce service dégradé envoie successivement trois *OUT-Contracts* correspondant soit à un même VPSN, soit à des VPSNs différents.

Selon notre étude, nous différencions deux types de maintenance des VSCUs : une maintenance interne effectuée par un service nommé *Internal VSCU Maintenance* et une autre externe effectuée par un service nommé *External VSCU Maintenance*. En effet, chaque service stocke dans son *Service Profile* les identifiants de ses Outer VSC-U&Ls, les identifiants des services responsables de ces communautés externes, ainsi que les adresses logiques de ces services responsables. Ainsi, chaque *OUT-Contract* sera envoyé par le service dégradé, en externe, vers tous les services responsables de ces Outer VSC-U&Ls, afin que les *External VSCU Maintenance* relatifs à leurs plates-formes interviennent et maintiennent ces Outer VSC-U&Ls. En parallèle, le service dégradé envoie aussi des *OUT-Contracts* en interne afin que l'*Internal VSCU Maintenance* intervient pour maintenir l'Inner VSC-U&L et la VSC-U&P correspondantes.

Nous commençons au début par la maintenance effectuée en externe. Pour cela, nous prenons le cas d'une plate-forme qui maintient l'Outer VSC-U&L associée à un service dégradé externe, ou autrement dit, nous prenons le cas d'une plate-forme qui maintient l'Inner VSC-U&L contenant un service externe qui est dégradé et qui a envoyé trois *OUT-Contracts* successifs. Pour ce faire, nous considérons une plate-forme qui a créé une Inner VSC-U&L, notée VSC-U&$L_{1,1}$, lors du déploiement d'un service $SE_{i1,j1,Px}$. Dans cette communauté, un service externe $SE_{i1,j2,Py}$, appartenant à un

fournisseur P_y, est sélectionné. Ce service considère cette communauté comme une de ses Outer VSC-U&Ls. Ainsi, quand l'agent de QoS du service $SE_{i1,j2,Py}$ détecte une dégradation de sa QoS ou un mauvais fonctionnement, il envoie un *OUT-Contract* au service $SE_{i1,j1,Px}$ responsable de cette Outer VSC-U&L. Ensuite, le service $SE_{i1,j1,Px}$ transmet chaque *OUT-Contract* reçu à l'*Events Manager* de sa plate-forme, tout en précisant son propre identifiant ainsi que celui du service $SE_{i1,j2,Py}$ dégradé. Au bout de trois *OUT-Contracts* successifs reçu pour le même service $SE_{i1,j2,Py}$, l'*Events Manager* décide qu'il faut lancer la maintenance de VSCUs. Or, puisqu'il y a deux identifiants précisés dans les *OUT-Contracts* successifs, l'*Events Manager* se rend compte que les *OUT-Contracts* proviennent d'un service externe dégradé, et décide ainsi de notifier le service *External VSCU Maintenance* pour que ce dernier maintienne l'Inner VSC-U&L ($\text{VSC-U\&L}_{1,1}$) contenant le service externe dégradé. Pour ce faire, l'*Events Manager* envoie vers le service *External VSCU Maintenance* la notification, *External VSCU Degradation*, qui contient le couple (Identifiant du service $SE_{i1,j2,Py}$ dégradé, Identifiant du service $SE_{i1,j1,Px}$ responsable de la VSC-U&L à gérer).

Comme le montre la Figure 8.6, le processus fonctionnel du service *External VSCU Maintenance* correspond à la séquence suivante d'opérations :

- Opération 1 : Le service *External VSCU Maintenance* envoie une requête à l'*Infoware* pour récupérer l'identifiant de l'Inner VSC-U&L relative au service $SE_{i1,j1,Px}$ interne (non dégradé).
- Opération 2 : Il envoie à l'*Infoware* une notification, *Remove Inner VSC-U&L*

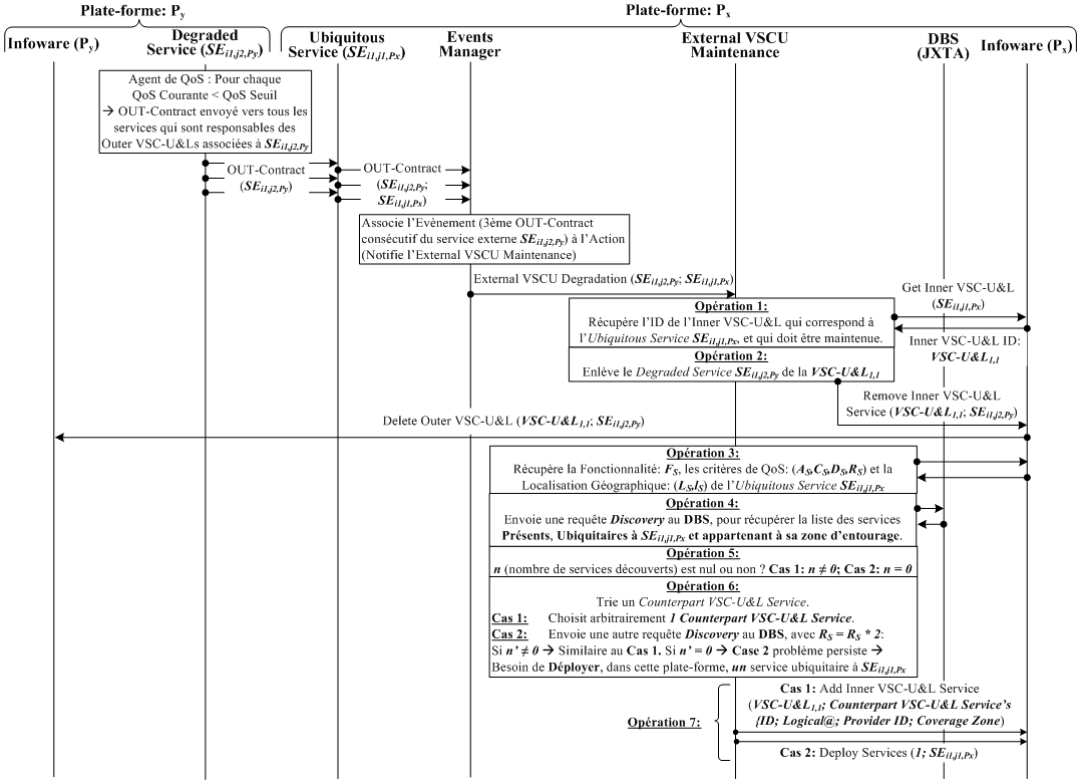

Fig. 8.6: Diagramme de séquence pour la maintenance externe des VSCUs.

Service, tout en précisant l'identifiant de l'Inner VSC-U&L à maintenir ainsi que l'identifiant du service $SE_{i1,j2,Py}$ externe dégradé. Ainsi, l'*Infoware* enlève ce service dégradé de l'Inner VSC-U&L, et il envoie une notification, *Delete Outer VSC-U&L*, vers le fragment d'*Infoware* relatif à la plate-forme du service dégradé. Ce fragment enlève du profil du service dégradé toute information sur son Outer VSC-U&L qui vient d'être maintenue.

- Opération 3 : Puisque le service *External VSCU Maintenance* a enlevé un service de l'Inner VSC-U&L, il doit le remplacer par un autre qui vérifie les caractéristiques de cette communauté. Pour ce faire, il commence par récupérer de l'*Infoware* la fonctionnalité, les quatre critères de QoS, la longitude et la latitude du service $SE_{i1,j1,Px}$ responsable de l'Inner VSC-U&L à maintenir.
- Opération 4 : Le service *External VSCU Maintenance* envoie une requête au service découverte DBS de JXTA. Ce dernier découvre la liste des services $\{SE_{i1,j,Pk} : j$ et k varient, avec $j \neq j1\}$ qui sont présents (disponibles, activables ou activés) et ubiquitaires au service $SE_{i1,j1,Px}$, et qui appartiennent à sa zone d'entourage géographique. Pour ce faire, le service DBS reçoit de la part du service *External VSCU Maintenance* les entrée suivantes : le rôle = "service" et la liste des critères de découverte = {la fonctionnalité du service $SE_{i1,j1,Px}$, ses quatre critères de QoS avec les quatre marges relatives, sa longitude, sa latitude, le rayon R_s de la zone d'entourage, ainsi que l'état disponible, activable ou activé}. Ensuite, le DBS envoie comme sortie la liste des identifiants des services découverts, leurs adresses logiques (SIP URIs) pour qu'ils puissent être contactés, leurs *Provider IDs* puisqu'ils sont multi-fournisseurs, ainsi que leurs zones de couverture qui seront utilisées dans la suite pour le handover sémantique.
- Opération 5 : Quand le service *External VSCU Maintenance* reçoit la réponse du DBS, il vérifie si le nombre de services découverts est nul ou pas.
- Opération 6 : Si le nombre est non nul, l'*External VSCU Maintenance* choisit arbitrairement un service. Si le nombre est nul, il relance la découverte avec le service DBS mais cette fois en multipliant par deux le rayon R_s de la zone d'entourage. Si après la deuxième découverte, le nombre de services découverts est devenu non nul, nous passons ainsi à une situation conforme au premier cas, et l'*External VSCU Maintenance* choisit arbitrairement un service.
- Opération 7 : Enfin, pour le cas où un service ubiquitaire est choisi pour remplacer le service dégradé, le service *External VSCU Maintenance* envoie à l'*Infoware* une notification, *Add Inner VSC-U&L Service*, tout en précisant l'identifiant de l'Inner VSC-U&L à maintenir, l'identifiant du service ubiquitaire choisi, son adresse logique, son *Provider ID* ainsi que sa zone de couverture. Pour le cas où même au bout de deux découvertes, le nombre de services découverts est toujours nul, l'*External VSCU Maintenance* envoie une notification, *Deploy Services*, à l'*Events Manager* afin de lancer le déploiement du service manquant. En effet, l'*External VSCU Maintenance* précise dans sa notification qu'il désire le déploiement d'un seul service et il indique aussi l'identifiant du service $SE_{i1,j1,Px}$ auquel le service à déployer doit être ubiquitaire. Une fois déployé, le service sera attaché par le service *VSCU Attachment* à l'Inner VSC-U&L en question.

Quand l'*Infoware* reçoit la notification, *Add Inner VSC-U&L Service*, il ajoute ce service à l'Inner VSC-U&L, et il envoie une notification vers le fragment d'*Infoware* relatif à la plate-forme du service choisi. Ainsi, ce fragment ajoute dans le profil de ce service l'identifiant de sa nouvelle Outer VSC-U&L qui vient d'être maintenue, ainsi que l'identifiant et l'adresse logique du service $SE_{i1,j1,Px}$ responsable de cette communauté. Il faut noter, que cette maintenance de l'Inner VSC-U&L s'effectue d'une manière transparente vis-à-vis des autres membres de cette communauté. Ainsi, aucune notification n'est envoyée vers leurs plates-formes.

Après avoir décrit le processus de maintenance des VSCUs par une plate-forme qui reçoit trois *OUT-Contracts* successifs envoyés par un service externe, nous traitons le cas de la maintenance des VSCUs par une plate-forme suite à la dégradation d'un service interne. Pour ce faire, nous prenons le même cas d'usage appliqué précédemment. Nous considérons que la plate-forme du fournisseur P_y a déjà créé l'Inner VSC-U&L, notée VSC-U&L$_{1,2}$, dont le responsable est le service $SE_{i1,j2,Py}$. Nous considérons aussi que ce dernier fait partie d'une VSC-U&P notée VSC-U&P$_{1,y}$. Nous supposons que le service $SE_{i1,j2,Py}$ se dégrade. Ainsi, l'agent de QoS envoie trois *OUT-Contracts* successifs en interne vers l'*Events Manager*. Ce dernier décide que c'est une dégradation interne puisqu'il n'y a qu'un seul identifiant de service dans les *OUT-Contracts*, et il envoie une notification, *Internal VSCU Degradation*, au service *Internal VSCU Maintenance* appartenant à sa plate-forme pour qu'il maintienne les différentes VSCUs (VSC-U&L$_{1,2}$ et VSC-U&P$_{1,y}$) relatives au service interne dégradé. La notification, *Internal VSCU Degradation*, contient seulement l'identifiant du service $SE_{i1,j2,Py}$ dégradé ; ce dernier est aussi le responsable de la VSC-U&L et de la VSC-U&P à gérer.

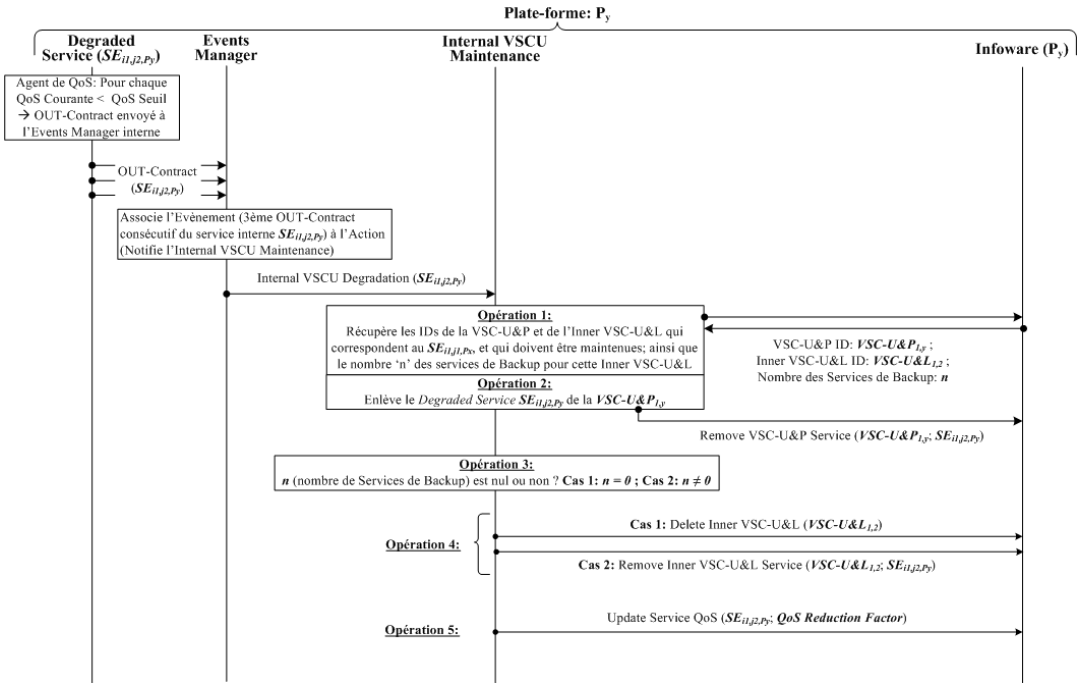

Fig. 8.7: Diagramme de séquence pour la maintenance interne des VSCUs.

Comme le montre la Figure 8.7, le processus fonctionnel du service *Internal VSCU Maintenance* correspond à la séquence suivante d'opérations :

- Opération 1 : Le service *Internal VSCU Maintenance* envoie une requête à l'*Infoware* pour récupérer l'identifiant de l'Inner VSC-U&L et la VSC-U&P relatives au service $SE_{i1,j2,Py}$ interne, ainsi que le nombre des services de *Backup* de cette Inner VSC-U&L.
- Opération 2 : Pour maintenir la VSC-U&P, il envoie à l'*Infoware* une notification, *Remove VSC-U&P Service*, tout en précisant l'identifiant de la VSC-U&P (VSC-U&P$_{1,y}$) à maintenir ainsi que l'identifiant du service $SE_{i1,j2,Py}$ interne dégradé. Ainsi, l'*Infoware* enlève ce service dégradé de la VSC-U&P, et il envoie la même notification, *Remove VSC-U&P Service*, vers tous les fragments d'*Infoware* relatifs aux plates-formes du fournisseur P_y. Cela permet à chacun de ces fragments de mettre à jour l'information concernant VSC-U&P$_{1,y}$ partagée entre eux.
- Opération 3 : Pour maintenir l'Inner VSC-U&L, le service *Internal VSCU Maintenance* vérifie si le nombre de services de *Backup* reçu est nul on non.
- Opération 4 : Si le nombre est nul, il envoie à l'*Infoware* une notification, *Delete Inner VSC-U&L*, tout en précisant l'identifiant de l'Inner VSC-U&L (VSC-U&L$_{1,2}$). Ainsi, l'*Infoware* supprime cette communauté et notifie les fragments d'*Infoware* concernés. Ces derniers suppriment eux aussi les informations qui concernent cette communauté qui a été considérée comme Outer VSC-U&L pour leurs services. Pour le deuxième cas, si le nombre de services de *Backup* pour l'Inner VSC-U&L n'est pas nul, le service *Internal VSCU Maintenance* envoie vers l'*Infoware* une notification, *Remove Inner VSC-U&L Service*, contenant l'identifiant de l'Inner VSC-U&L (VSC-U&L$_{1,2}$) à maintenir ainsi que l'identifiant du service $SE_{i1,j2,Py}$ interne dégradé. Ainsi, l'*Infoware* enlève le service interne dégradé de cette VSC-U&L. Or, puisque le service enlevé était le responsable de cette communauté, l'*Infoware* sélectionne parmi les services de *Backup* un nouveau responsable pour cette communauté. Ensuite, une notification est envoyée par l'*Infoware* vers tous les fragments d'*Infoware* relatifs aux différents services contenus dans la VSC-U&L$_{1,2}$, afin que chaque fragment mette à jour l'identifiant et l'adresse logique du nouveau responsable de la communauté VSC-U&L$_{1,2}$.
- Opération 5 : Enfin, le service *Internal VSCU Maintenance* se rend compte qu'au bout de trois *OUT-Contract* successifs pour un même service dégradé, la QoS de conception à laquelle les utilisateurs sont abonnés ne reflète plus le comportement de ce service. Pour cela, l'*Internal VSCU Maintenance* décide de réduire cette valeur de conception et par conséquent la valeur seuil à ne pas dépasser. Pour cela, il envoie à l'*Infoware* une autre notification, *Update Service QoS*, contenant l'identifiant $SE_{i1,j2,Py}$ dégradé ainsi que le facteur de réduction par lequel doivent être multipliées les anciennes valeurs de QoS.

Ainsi, l'*Infoware* met à jour les valeurs de QoS dans le *Service Profile* de $SE_{i1,j2,Py}$. Suite à cette mise à jour de la QoS de conception, le service *VSCU Attachment* sera notifié puisqu'il est aussi abonné à ce type d'évènement. Il traite le service adapté comme un service déployé. Par conséquent, chaque service adapté aura une nouvelle Inner VSC-U&L et une nouvelle VSC-U&P. Afin d'empêcher les deux premières notifications

envoyées par l'*Internal VSCU Maintenance* à l'*Infoware* de modifier les nouvelles VSCUs attribuées au service $SE_{i1,j2,Py}$, l'*Infoware* vérifie avant de mettre à jour le *Service Profile* de $SE_{i1,j2,Py}$, que ces deux premières notifications sont déjà exécutées.

En plus de la maintenance effectuée en externe pour les Outer VSC-U&Ls du service dégradé, ainsi que la maintenance en interne de sa VSC-U&P et de son Inner VSC-U&L dont il est responsable, un autre cas de maintenance est aussi possible : c'est la maintenance de l'Inner VSC-U&L attachée au service dégradé dans le cas où ce service n'est plus le responsable mais un simple service de *Backup*. Pour résoudre ce cas, nous proposons que le service dégradé envoie aussi un *OUT-Contract* au service responsable de son Inner VSC-U&L. Par conséquent, nous procédons comme dans le cas de la maintenance en externe effectuée par le service *External VSCU Maintenance*.

8.1.4 Mise en œuvre des VSCUs

Afin de mettre en exergue l'approche évènementielle et distribuée que nous proposons pour la gestion des communautés virtuelles, nous présentons dans cette sous-section une mise en œuvre organisationnelle de notre solution. Nous nous contentons de décrire, d'après les Figures 8.8 et 8.9, les cas suivants : la création de VSC-U&Ls, la création d'une VSC-U&P, l'attachement à une VSC-U&P, et la maintenance externe des VSC-U&Ls. Dans la suite, l'approche évènementielle est représentée par un ensemble

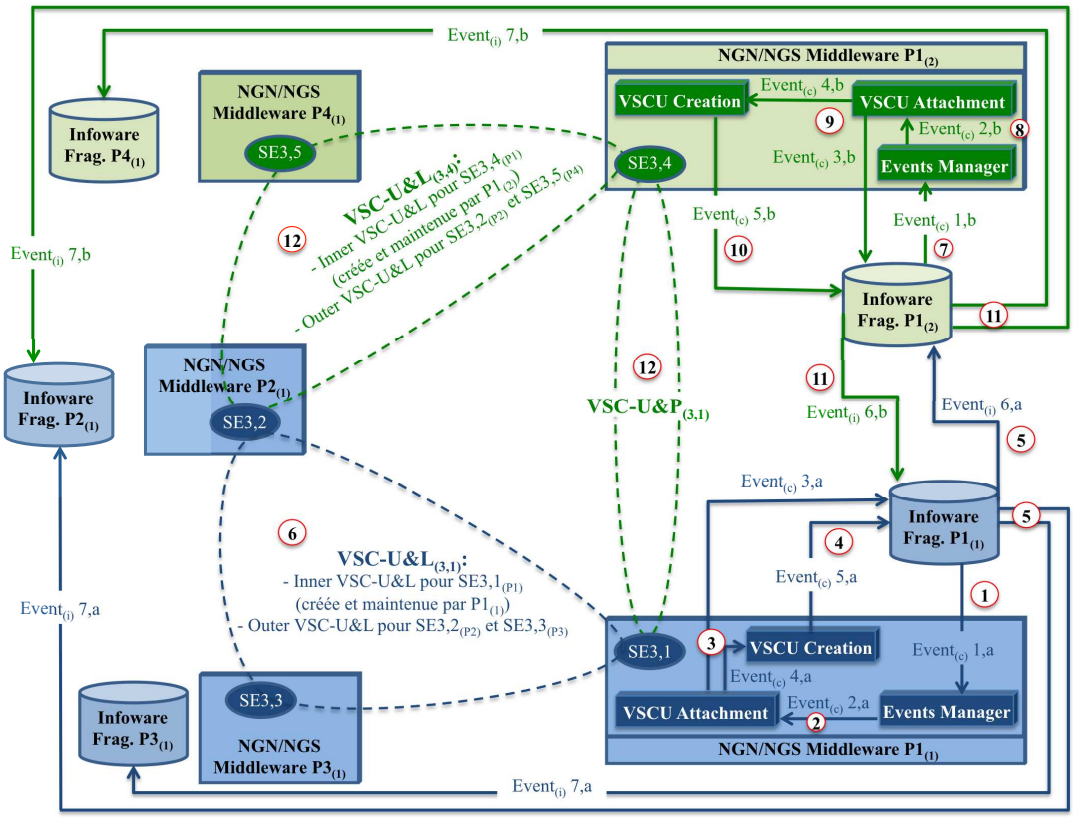

Fig. 8.8: Mise en œuvre de l'attachement et de la création des VSCUs.

d'évènements classifiés sous deux types :

- *Event*$_{(c)}$: il représente un *Computing Event* échangé entre un service et l'*Infoware*, ou entre deux services.
- *Event*$_{(i)}$: il représente un *Informational Event* échangé entre deux fragments d'*Infoware* distribués.

Nous considérons, comme le montre la Figure 8.8, un fournisseur de service *P1* ayant deux plates-formes distribuées (*P1*$_{(1)}$ et *P1*$_{(2)}$). Dans *P1*$_{(1)}$, ce fournisseur désire déployer un nouveau élément de service noté *SE3,1*$_{(P1)}$ (la notation *SEi,j*$_{Pk}$ correspond à l'identifiant du service ; le "i" représente la fonctionnalité du service, le "j" représente le "j^{eme}" service ubiquitaire de type *SEi*, et "Pk" représente un "k^{eme}" fournisseur de services). Lors du déploiement de *SE3,1*$_{(P1)}$, le fragment d'*Infoware* associé à la plate-forme *P1*$_{(1)}$ crée un *Service Profile* et envoie à l'*Events Manager* de *P1*$_{(1)}$ un premier évènement noté *Event*$_{(c)}$ *1,a* (où "a" signifie que cet évènement correspond à la partie "a" de la figure, c.à.d. à la plate-forme *P1*$_{(1)}$).

Event$_{(c)}$ *1,a : Service Deployment Notification (SE3,1*$_{(P1)}$*)*

L'*Events Manager* associe cet évènement à une action qui consiste à notifier le service *VSCU Attachment* de *P1*$_{(1)}$, en transmettant un évènement *Event*$_{(c)}$ *2,a*.

Event$_{(c)}$ *2,a : Service Deployment Notification (SE3,1*$_{(P1)}$*)*

Le *VSCU Attachment* n'arrive pas à attacher le service *SE3,1*$_{(P1)}$ à aucune Inner VSC-U&L ou VSC-U&P existantes. Ainsi, il décide qu'il faut créer ces communautés. Pour ce faire, il envoie à l'*Infoware* une notification *Event*$_{(c)}$ *3,a* afin que ce dernier crée la VSC-U&P qui soit relative au service *SE3,1*$_{(P1)}$ et qui le contient tout seul. En outre, afin de créer l'Inner VSC-U&L correspondant au service déployé *SE3,1*$_{(P1)}$, le *VSCU Attachment* envoie un évènement *Event*$_{(c)}$ *4,a* au service *VSCU Creation*.

Event$_{(c)}$ *3,a : Create VSC-U&P (SE3,1*$_{(P1)}$*)*

Event$_{(c)}$ *4,a : Create VSC-U&L (SE3,1*$_{(P1)}$*)*

Ensuite, le *VSCU Creation* découvre deux services *SE3,2*$_{(P2)}$ et *SE3,3*$_{(P3)}$ appartenant respectivement à la plate-forme *P2*$_{(1)}$ d'un fournisseur P_2 et à la plate-forme *P3*$_{(1)}$ d'un fournisseur P_3. Ces deux services sont présents, ubiquitaires à *SE3,1*$_{(P1)}$ et se trouvent dans sa zone d'entourage. Nous supposons que "m", le nombre minimal de services dans une VSC-U&L, est égale à 2. Ainsi, nous n'avons pas besoin de déployer de nouveaux services ubiquitaires dans *P1*$_{(1)}$ pour compléter la communauté VSC-U&L. Par conséquent, le *VSCU Creation* décide de créer l'Inner VSC-U&L qui est relative au service *SE3,1*$_{(P1)}$ et qui contient les deux services *SE3,2*$_{(P2)}$ et *SE3,3*$_{(P3)}$. Pour ce faire, il envoie à l'*Infoware* une notification *Event*$_{(c)}$ *5,a* contenant l'identifiant du service déployé ainsi que les identifiants des services ubiquitaires découverts, leurs adresses logiques, leurs *Provider IDs* et leurs zones de couverture.

$Event_{(c)}$ 5,a : Inner VSC-U&L Creation (SE3,1$_{(P1)}$; SE3,2$_{(P2)}$; SE3,3$_{(P3)}$; Log@3,2 ; Log@3,3 ; P_2 ; P_3 ; CZ3,2$_{(P2)}$; CZ3,3$_{(P3)}$)

Une fois le fragment d'*Infoware* relatif à la plate-forme $P1_{(1)}$ reçoit les évènements $Event_{(c)}$ *3,a* et $Event_{(c)}$ *5,a*, il crée les profils de la VSC-U&P$_{(3,1)}$ et de la VSC-U&L$_{(3,1)}$. En effet, pour la VSC-U&P$_{(3,1)}$, le couple (3,1) représente respectivement le type de service (*SE3*) et le type de fournisseur (*P1*). Pour la VSC-U&L$_{(3,1)}$, le couple (3,1) représente respectivement le type de service (*SE3*) et la zone d'entourage du service *SE3,1*$_{(P1)}$. Après avoir créé ces profils, ce fragment d'*Infoware* met à jour les autres fragments d'*Infoware* concernés. Il envoie une notification $Event_{(i)}$ *6,a* au fragment d'*Infoware* associé à la plate-forme $P1_{(2)}$ puisque cette dernière est une des plates-formes du fournisseur P_1. Dans cet évènement, il envoie tout le profil de la VSC-U&P créée ainsi que l'identifiant de *SE3,1*$_{(P1)}$, son adresse logique et sa zone de couverture.

$Event_{(i)}$ 6,a : VSC-U&P Creation (VSC-U&P Profile ; SE3,1$_{(P1)}$; Log@3,1 ; CZ3,1$_{(P1)}$)

De plus, le fragment d'*Infoware* associé à $P1_{(1)}$ envoie des notifications $Event_{(i)}$ *7,a* aux fragments d'*Infoware* associés aux plates-formes $P2_{(1)}$ et $P3_{(1)}$ puisqu'elles contiennent les deux services *SE3,2*$_{(P2)}$ et *SE3,3*$_{(P3)}$ introduits dans la VSC-U&L$_{(3,1)}$. Dans cet évènement, le fragment d'*Infoware* précise l'identifiant de VSC-U&L$_{(3,1)}$ ainsi que l'identifiant et l'adresse logique de son service responsable *SE3,1*$_{(P1)}$. Ainsi, ces informations envoyées seront associées aux profils des services *SE3,2*$_{(P2)}$ et *SE3,3*$_{(P3)}$, qui considèrent la VSC-U&L$_{(3,1)}$ créée comme une Outer VSC-U&L.

$Event_{(i)}$ 7,a : Outer VSC-U&L Creation (VSC-U&L$_{(3,1)}$; SE3,1$_{(P1)}$; Log@3,1)

A la fin de cette partie "a" concernant la plate-forme $P1_{(1)}$, nous obtenons une VSC-U&P$_{(3,1)}$ contenant le service *SE3,1*$_{(P1)}$, et une VSC-U&L$_{(3,1)}$ qui est Inner VSC-U&L pour *SE3,1*$_{(P1)}$ et Outer VSC-U&L pour *SE3,2*$_{(P2)}$ et *SE3,3*$_{(P3)}$. Par conséquent, c'est la plate-forme $P1_{(1)}$ qui sera responsable de la maintenance de cette VSC-U&L$_{(3,1)}$.

Selon la Figure 8.8, en parallèle au déploiement du service *SE3,1*$_{(P1)}$ dans la plate-forme $P1_{(1)}$ (Partie "a"), le fournisseur P_1 désire aussi déployer, dans sa deuxième plate-forme $P1_{(2)}$, un service *SE3,4*$_{(P1)}$ qui est ubiquitaire à *SE3,1*$_{(P1)}$. Ainsi, le fragment d'*Infoware* associé à la plate-forme $P1_{(2)}$ crée un *Service Profile* et envoie à l'*Events Manager* de $P1_{(2)}$ un premier évènement noté $Event_{(c)}$ *1,b* (où "b" signifie que cet évènement correspond à la partie "b" de la figure, c.à.d. à la plate-forme $P1_{(2)}$). Ce dernier notifie le service *VSCU Attachment* de $P1_{(2)}$, en transmettant l'évènement $Event_{(c)}$ *2,b*.

$Event_{(c)}$ 1,b : Service Deployment Notification (SE3,4$_{(P1)}$)

$Event_{(c)}$ 2,b : Service Deployment Notification (SE3,4$_{(P1)}$)

Ensuite, le *VSCU Attachment* décide d'attacher le service $SE3{,}4_{(P1)}$ à la VSC-U&P$_{(3,1)}$ déjà créée. Pour cela, il envoie à son fragment d'*Infoware* un évènement $Event_{(c)}$ *3,b* dans lequel il précise l'identifiant du service attaché ainsi que l'identifiant de VSC-U&P$_{(3,1)}$. En outre, le *VSCU Attachment* n'arrive pas à attacher le service $SE3{,}1_{(P1)}$ à aucune Inner VSC-U&L. Pour cela, il envoie une notification $Event_{(c)}$ *4,b* au service *VSCU Creation* afin que ce dernier crée l'Inner VSC-U&L correspondant au service déployé $SE3{,}4_{(P1)}$.

$Event_{(c)}$ 3,b : Attach VSC-U&P Service ($SE3{,}4_{(P1)}$; VSC-U&P$_{(3,1)}$)

$Event_{(c)}$ 4,b : Create VSC-U&L ($SE3{,}4_{(P1)}$)

Le service *VSCU Creation* lance son processus fonctionnel et découvre deux services $SE3{,}2_{(P2)}$ et $SE3{,}5_{(P4)}$ qui appartiennent respectivement à la plate-forme $P2_{(1)}$ d'un fournisseur P_2 et à la plate-forme $P4_{(1)}$ d'un fournisseur P_4. Ces deux services sont présents, ubiquitaires à $SE3{,}4_{(P1)}$ et se trouvent dans sa zone d'entourage. Par conséquent, le service *VSCU Creation* décide de créer l'Inner VSC-U&L qui est relative au service $SE3{,}4_{(P1)}$ et qui contient les deux services $SE3{,}2_{(P2)}$ et $SE3{,}5_{(P4)}$. Pour ce faire, il envoie à l'*Infoware* une notification $Event_{(c)}$ *5,b*.

$Event_{(c)}$ 5,b : Inner VSC-U&L Creation ($SE3{,}4_{(P1)}$; $SE3{,}2_{(P2)}$; $SE3{,}5_{(P4)}$; Log@3,2 ; Log@3,5 ; P_2 ; P_4 ; $CZ3{,}2_{(P2)}$; $CZ3{,}5_{(P4)}$)

Une fois le fragment d'*Infoware* relatif à la plate-forme $P1_{(2)}$ reçoit les évènements $Event_{(c)}$ *3,b* et $Event_{(c)}$ *5,b*, il adapte le profil de la VSC-U&P$_{(3,1)}$ en ajoutant le service $SE3{,}4_{(P1)}$, et il crée le profil de la VSC-U&L$_{(3,4)}$. Ensuite, il met à jour les autres fragments d'*Infoware* concernés. Il envoie une notification $Event_{(i)}$ *6,b* au fragment d'*Infoware* associé à la plate-forme $P1_{(1)}$ puisque cette dernière fait partie des plates-formes du même fournisseur P_1. Dans cet évènement, il n'a pas besoin d'envoyer tout le profil du VSC-U&P créé, puisque ce profil existe déjà dans les autres fragments d'*Infoware*. Ainsi, il suffit d'envoyer l'identifiant de la VSC-U&P à adapter, ainsi que l'identifiant du service $SE3{,}4_{(P1)}$ attaché, son adresse logique et sa zone de couverture.

$Event_{(i)}$ 6,b : VSC-U&P Service Attachment (VSC-U&P$_{(3,1)}$; $SE3{,}4_{(P1)}$; Log@3,4 ; $CZ3{,}4_{(P1)}$)

Le fragment d'*Infoware* associé à $P1_{(1)}$ envoie des notifications $Event_{(i)}$ *7,b* aux fragments d'*Infoware* associés aux plates-formes $P2_{(1)}$ et $P4_{(1)}$ contenant les deux services $SE3{,}2_{(P2)}$ et $SE3{,}5_{(P4)}$ introduits dans la VSC-U&L$_{(3,4)}$. Ainsi, $SE3{,}2_{(P2)}$ et $SE3{,}3_{(P3)}$ considèrent la VSC-U&L$_{(3,4)}$ créée comme une Outer VSC-U&L.

$Event_{(i)}$ 7,b : Outer VSC-U&L Creation (VSC-U&L$_{(3,4)}$; $SE3{,}4_{(P1)}$; Log@3,4)

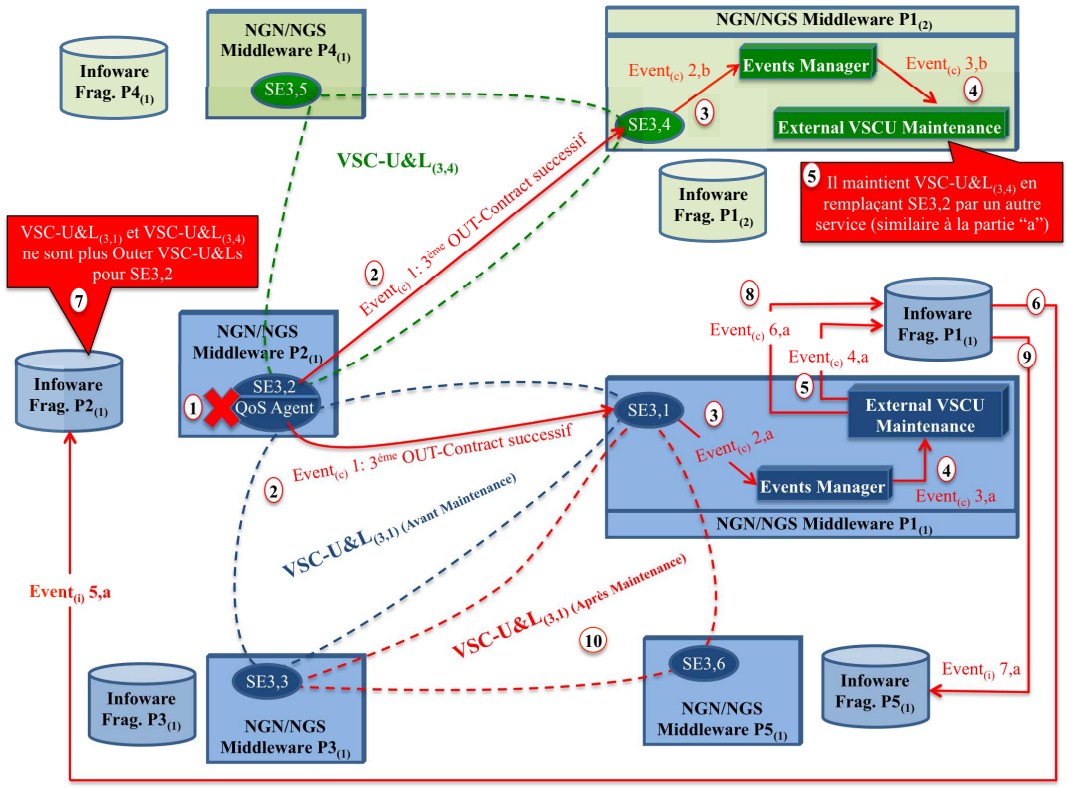

Fig. 8.9: Mise en œuvre de la maintenance externe des VSCUs.

A la fin de cette phase de création et d'attachement, nous obtenons les communautés virtuelles suivantes : une VSC-U&P$_{(3,1)}$ contenant les services *SE3,1*$_{(P1)}$ et *SE3,4*$_{(P1)}$; une VSC-U&L$_{(3,1)}$ qui est Inner VSC-U&L pour *SE3,1*$_{(P1)}$ et Outer VSC-U&L pour *SE3,2*$_{(P2)}$ et *SE3,3*$_{(P3)}$; ainsi qu'une VSC-U&L$_{(3,4)}$ qui est Inner VSC-U&L pour *SE3,4*$_{(P1)}$ et Outer VSC-U&L pour *SE3,2*$_{(P2)}$ et *SE3,5*$_{(P4)}$. Par conséquent, la plate-forme *P1*$_{(1)}$ sera responsable de la maintenance de la VSC-U&L$_{(3,1)}$, quand à la plate-forme *P1*$_{(2)}$, elle sera responsable de la maintenance de la VSC-U&L$_{(3,4)}$. En outre, nous constatons que le service *SE3,2*$_{(P2)}$ participe à deux Outer VSC-U&Ls.

Nous décrivons maintenant le cas d'usage de la Figure 8.9 qui traite la phase de maintenance externe des VSCUs créées. Nous considérons que l'agent de QoS du service *SE3,2*$_{(P2)}$ détecte, pour trois fois consécutives, une dégradation de sa QoS courante. Ainsi, un processus de maintenance interne au niveau de la plate-forme *P2*$_{(1)}$ doit être exécuté afin de maintenir l'Inner VSC-U&L du service *SE3,2*$_{(P2)}$ ainsi que la VSC-U&P qui lui est associée. En outre, un processus de maintenance externe doit aussi s'effectuer afin de maintenir les Outer VSC-U&Ls associées à *SE3,2*$_{(P2)}$. Cependant, dans notre cas de figure, nous désirons montrer l'approche distribuée que nous adoptons pour la maintenance des VSCUs. Pour cette raison, nous décrivons seulement la partie de maintenance externe distribuée entre les différentes plates-formes.

Suite à la phase de création des VSCUs, le service *SE3,2*$_{(P2)}$ fut associé à deux Outer VSC-U&Ls : la première est la VSC-U&L$_{(3,1)}$ dont le responsable est le service *SE3,1*$_{(P1)}$, et la deuxième est la VSC-U&L$_{(3,4)}$ dont le responsable est le service

SE3,4$_{(P1)}$. Pour cela, après chaque dégradation, l'agent de QoS associé à *SE3,2*$_{(P2)}$ envoie un *OUT-Contract* à tous les services responsables de ses Outer VSC-U&Ls. Par conséquent, il envoie des *OUT-Contracts* contenant l'identifiant de *SE3,2*$_{(P2)}$ vers les services *SE3,1*$_{(P1)}$ et *SE3,4*$_{(P1)}$. Ces derniers transmettent chacun l'*OUT-Contract* reçu à son *Events Manager*, tout en ajoutant chacun son propre identifiant. Ensuite, au bout de la troisième dégradation, l'agent de QoS de *SE3,2*$_{(P2)}$ envoie un troisième *OUT-Contract*, sous la forme d'un évènement *Event*$_{(c)}$ *1*, vers les services *SE3,1*$_{(P1)}$ et *SE3,4*$_{(P1)}$. Ces derniers transmettent les *OUT-Contracts* vers leurs *Events Managers*, sous la forme d'un évènement *Event*$_{(c)}$ *2,a* et d'un évènement *Event*$_{(c)}$ *2,b*.

Event$_{(c)}$ *1 : OUT-Contract (SE3,2*$_{(P2)}$*)*

Event$_{(c)}$ *2,a : OUT-Contract (SE3,2*$_{(P2)}$ *; SE3,1*$_{(P1)}$*)*

Event$_{(c)}$ *2,b : OUT-Contract (SE3,2*$_{(P2)}$ *; SE3,4*$_{(P1)}$*)*

Par conséquent, chaque *Events Manager* décide qu'il faut notifier le service *External VSCU Maintenance* appartenant à sa plate-forme, afin qu'il puisse maintenir les Inner VSC-U&Ls qui contiennent le service SE3,2$_{(P2)}$. Pour cela, les évènements *Event*$_{(c)}$ *3,a* et *Event*$_{(c)}$ *3,b* sont envoyés en parallèle.

Event$_{(c)}$ *3,a : External VSCU Degradation (SE3,2*$_{(P2)}$ *; SE3,1*$_{(P1)}$*)*

Event$_{(c)}$ *3,b : External VSCU Degradation (SE3,2*$_{(P2)}$ *; SE3,4*$_{(P1)}$*)*

Ensuite, chaque *External VSCU Maintenance* lance son processus de maintenance pour son Inner VSC-U&L correspondante. Dans la suite, nous nous contentons de décrire le processus de maintenance de la VSC-U&L$_{(3,1)}$, effectué par l'*External VSCU Maintenance* de la plate-forme *P1*$_{(1)}$. En effet, la maintenance de la VSC-U&L$_{(3,4)}$ sera effectuée par l'*External VSCU Maintenance* de la plate-forme *P1*$_{(2)}$ d'une façon similaire. Pour maintenir la VSC-U&L$_{(3,1)}$, l'*External VSCU Maintenance* commence par enlever le service *SE3,2*$_{(P2)}$ de cette communauté. Pour cela, il envoie une notification *Event*$_{(c)}$ *4,a* à son fragment d'*Infoware* tout en précisant l'identifiant de la communauté ainsi que l'identifiant du service à enlever. Par conséquent, ce fragment d'*Infoware* envoie à son tour une notification, *Event*$_{(i)}$ *5,a*, vers le fragment d'*Infoware* qui est associé à la plate-forme *P2*$_{(1)}$ où se trouve le service *SE3,2*$_{(P2)}$. Suite à cette notification, toute information qui concerne VSC-U&L$_{(3,1)}$ est enlevée du profil de *SE3,2*$_{(P2)}$. Ainsi, ce dernier ne considère plus VSC-U&L$_{(3,1)}$ comme son Outer VSC-U&L.

Event$_{(c)}$ *4,a : Remove Inner VSC-U&L Service (VSC-U&L*$_{(3,1)}$ *; SE3,2*$_{(P2)}$*)*

Event$_{(i)}$ *5,a : Delete Outer VSC-U&L (VSC-U&L*$_{(3,1)}$ *; SE3,2*$_{(P2)}$*)*

En parallèle, l'*External VSCU Maintenance* continue son processus de gestion de la VSC-U&L$_{(3,1)}$. Pour cela, il s'appuie sur le service DBS pour découvrir un service ubiquitaire à SE3,2$_{(P2)}$ pour le remplacer. Nous supposons dans notre cas de figure que la découverte a mené à trouver un service SE3,6$_{(P5)}$ appartenant à une plate-forme $P5_{(1)}$ d'un fournisseur P_5. Ce service est présent, ubiquitaire à SE3,2$_{(P2)}$ et se trouve dans sa zone d'entourage. Dans ce cas là, nous n'avons pas besoin de déployer un service ubiquitaire en interne afin qu'il remplace SE3,2$_{(P2)}$. Pour effectuer le remplacement, le service *External VSCU Maintenance* envoie ainsi une notification *Event$_{(c)}$ 6,a* à son fragment d'*Infoware* tout en précisant l'identifiant de la communauté ainsi que l'identifiant du service à ajouter, son adresse logique, son *Provider ID* et sa zone de couverture. Par conséquent, ce fragment d'*Infoware* envoie à son tour une notification, *Event$_{(i)}$ 7,a*, vers le fragment d'*Infoware* qui est associé à la plate-forme $P5_{(1)}$ où se trouve le service *SE3,6$_{(P5)}$*. Ainsi, ce fragment ajoute, dans le profil de *SE3,6$_{(P5)}$*, l'identifiant de sa nouvelle Outer VSC-U&L qui vient d'être maintenue, ainsi que l'identifiant et l'adresse logique du service *SE3,1$_{(P1)}$* qui est responsable pour cette communauté. En effet, puisque la maintenance de l'Inner VSC-U&L s'effectue d'une manière transparente vis-à-vis des autres membres de cette communauté, aucune notification n'est envoyée vers la plate-forme $P3_{(1)}$ où se trouve le service *SE3,3$_{(P3)}$*.

Event$_{(c)}$ 6,a : Add Inner VSC-U&L Service (VSC-U&L$_{(3,1)}$; SE3,6$_{(P5)}$; Log@3,6 ; P_5 ; CZ3,6$_{(P5)}$)

Event$_{(i)}$ 7,a : Outer VSC-U&L Creation (VSC-U&L$_{(3,1)}$; SE3,1$_{(P1)}$; Log@3,1)

Enfin, nous obtenons une nouvelle VSC-U&L$_{(3,1)}$ qui reste Inner VSC-U&L pour *SE3,1$_{(P1)}$*, mais qui devient Outer VSC-U&L pour *SE3,3$_{(P3)}$* et *SE3,6$_{(P5)}$*. Par conséquent, cette communauté reste toujours maintenue par la plate-forme $P1_{(1)}$ où se trouve son service responsable *SE3,1$_{(P1)}$*.

D'après cette mise en œuvre organisationnelle, nous avons montré comment les différents services et plates-formes distribués collaborent pour créer et gérer les différents types de communautés virtuelles. De plus, nous avons montré comment les informations en commun sont partagées sans duplication lourde, et comment les mises à jour sont effectuées avec des notifications simples qui ne chargent pas trop le réseau de transport.

Pour conclure, dans cette section nous avons proposé une solution qui anticipe dynamiquement la coupure de la session en remplaçant les services pré-provisionnés dans le VPSN de l'utilisateur par d'autres services ubiquitaires. Cette solution est basée sur le concept des communautés virtuelles et elle garantie d'une manière transparente et dynamique la continuité de la session de bout-en-bout tout en tenant compte des préférences fonctionnelles et non-fonctionnelles (critères de QoS) de l'utilisateur. Cependant, la question qui se pose est : comment adapter le VPSN aux changements dans le contexte ambiant de l'utilisateur, tout au long de sa mobilité spatiale et temporelle ? Pour répondre à cette question, nous proposons, dans la section suivante, notre deuxième solution pour la gestion de la continuité de services, qui est basée sur le concept de handover sémantique.

8.2 Le Handover Sémantique

8.2.1 Description du concept

Selon notre approche *user-centric*, l'utilisateur désire avoir un accès continu à ses services pour toute commutation de terminal (mobilité de l'utilisateur) ou de point d'accès réseau (mobilité du terminal). Suite à ce contexte mobile, l'utilisateur subit une forte instabilité dans son contexte ambiant. De nos jours, le concept de réseaux ambiants permet de prendre en considération le nouveau contexte ambiant de l'utilisateur. De plus, les mécanismes de handover permettent de gérer la mobilité au niveau réseau et de maintenir l'accès de l'utilisateur à ses services. Cependant, toutes ces solutions ne prennent pas en considération la couche service, et négligent deux points essentiels : le premier est celui de l'influence de la latence sur la QoE de l'utilisateur ; et le deuxième est celui du besoin d'étendre la prise en conscience du contexte ambiant vers la couche service, afin d'obtenir une session de services ambiants.

Pour dépasser ces défaillances de la gestion de mobilité au niveau réseau, nous proposons le concept de handover sémantique. Ce dernier est une extension des mécanismes de handover des couches réseaux vers la couche service. En effet, tout au long de la mobilité spatiale (mobilité de l'utilisateur ou mobilité du terminal) et temporelle de l'utilisateur, de nouveaux services ambiants sont découverts. Ces derniers peuvent contenir des services nouveaux ainsi que des services ubiquitaires à ceux pré-provisionnés dans la session de services (VPSN) de l'utilisateur. Par conséquent, afin de diminuer la latence et d'améliorer la QoE et la prise en conscience du contexte ambiant au niveau service, le mécanisme de handover sémantique propose d'adapter, d'une manière continue et transparente, le VPSN de l'utilisateur à son nouveau contexte ambiant.

D'une part, ce nouveau mécanisme est sémantique puisqu'il consiste à remplacer un service par un autre ayant la même fonctionnalité et appartenant au nouveau contexte ambiant de l'utilisateur. D'autre part, le handover sémantique ne se limite pas à l'aspect fonctionnel du service, mais il tient aussi compte des préférences non fonctionnelles de l'utilisateur, c.à.d. la QoS demandée. Pour ce faire, il exécute le handover entre des services ubiquitaires qui sont non seulement fonctionnellement équivalents mais aussi QoS équivalents. Par conséquent, nous améliorons la QoE de l'utilisateur et sa prise en conscience de son nouveau contexte ambiant, en le connectant aux services les plus proches et les plus adaptés à ses préférences (fonctionnelles et non fonctionnelles).

Afin de supporter notre architecture EDA, nous basons notre handover sémantique sur une approche évènementielle. Ainsi, suite à des évènements liés à la mobilité, le processus de handover s'initie, prend des décisions et enfin s'exécute pour adapter le VPSN de l'utilisateur à son nouveau contexte ambiant. Pour ce faire, le handover sémantique s'appuie, par analogie aux mécanismes de handover appliqués au niveau réseau, sur trois rôles dont chacun peut être joué par un ou plusieurs acteurs. Nous identifions les rôles suivants :

- Initiateur : il détecte un évènement de mobilité spatiale et initie par la suite le processus de handover sémantique en envoyant une notification au décideur.

- Décideur : il prend la décision dans un handover sémantique en se basant sur des informations décisionnelles temps réel. Ensuite, il transmet la décision à l'exécuteur.
- Exécuteur : il exécute la décision de handover sémantique et effectue les adaptations dans le VPSN.

Cette approche évènementielle adoptée par le handover sémantique permet d'améliorer la capacité de réaction et d'adaptation dans le système de gestion suite à une mobilité spatiale de la part de l'utilisateur.

Après avoir décrit le concept de handover sémantique, nous présentons dans la suite : premièrement, son aspect architectural qui correspond à une structure distribuée à base de zones de couverture, et deuxièmement, l'aspect fonctionnel qui correspond à l'ensemble des acteurs participant à ce handover sémantique.

8.2.2 Aspect architectural du Handover Sémantique

Afin de pouvoir appliquer le concept de handover sémantique, nous basons sa dimension architecturale sur une hypothèse qui consiste à avoir une distribution à grande échelle des services ubiquitaires. En effet, nous considérons que chaque fournisseur de services, selon sa stratégie économique sur le marché des services et selon les heuristiques représentant la répartition géographique des charges d'accès aux services, doit distribuer ses services ubiquitaires de façon à couvrir une zone géographique bien déterminée. L'objectif est de répondre aux demandes du plus grand nombre d'utilisateurs, en les reliant toujours aux services qui répondent le mieux à leurs contextes ambiants. Des contrats de *peering* doivent aussi être négociés entre les fournisseurs qui offrent des services ubiquitaires, afin de se mettre d'accord sur une politique de paiement lorsqu'ils connectent leurs utilisateurs, d'une manière transparente et tout au long de la mobilité temporelle et spatiale de ces derniers, aux sources de services les plus proches. Ainsi, la question qui se pose est la suivante : quelle est la meilleure approche à adopter pour la distribution des services ubiquitaires afin d'appliquer le processus de handover sémantique de la façon la plus efficace, et garantir par la suite une adaptation temps réel pour le VPSN de l'utilisateur mobile ?

Pour répondre à cette question, nous avons étudié le mécanisme utilisé par les opérateurs, au niveau des réseaux d'accès cellulaires, pour distribuer efficacement leurs antennes d'accès. En effet, les opérateurs considèrent que chaque point d'accès est responsable pour une certaine zone de couverture. Ainsi, le handover au niveau réseau s'effectue quand le terminal se déplace entre ces zones de couverture. La détection et l'initiation de ce handover réseau vers un nouveau point d'accès s'appuie sur une comparaison entre les puissances des différents signaux reçus. Afin de profiter de la réussite et de l'évolution rapide au niveau réseau, nous proposons d'étendre aussi la technique de distribution des réseaux d'accès cellulaires à la couche service. Par conséquent, chaque fournisseur de services distribue ses propres services ubiquitaires, tout en attribuant à chacun d'entre eux une responsabilité sur une zone de couverture bien déterminée. La zone de couverture d'un service donné correspond à un cercle avec un certain rayon et ayant comme centre l'adresse géographique du service en question.

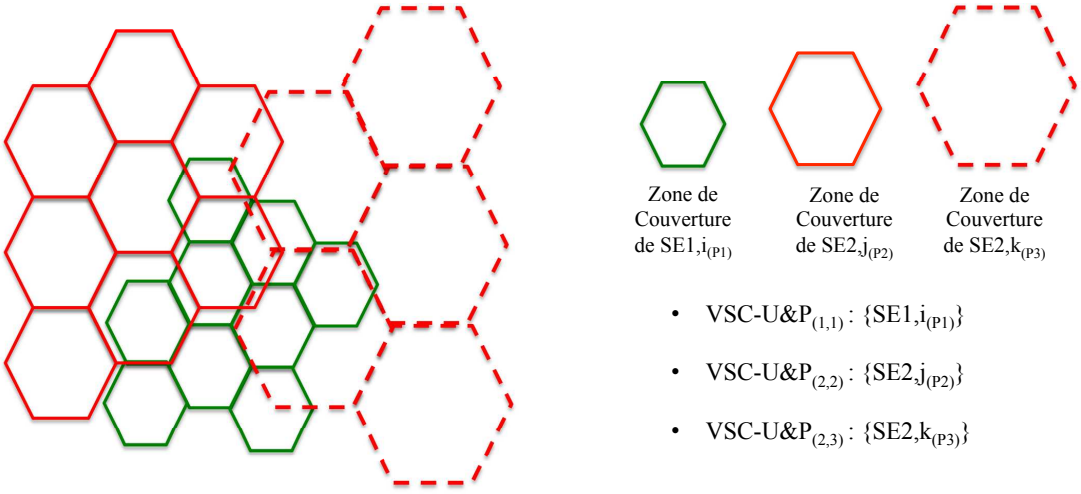

Fig. 8.10: Approche architecturale distribuée du Handover Sémantique.

Pour représenter cet aspect architectural basé sur les zones de couvertures distribuées, nous prenons, selon la Figure 8.10, le cas de trois fournisseurs de services (P_1, P_2, P_3) qui ont distribué leurs services ubiquitaires respectifs ($\{SE1,i_{(P1)}\}$, $\{SE2,j_{(P2)}\}$, $\{SE2,k_{(P3)}\}$) sur différentes zones de couvertures. En effet, chaque fournisseur associe une zone de couverture à chacun de ses services ubiquitaires. Pour des raisons de clarté, nous représentons une zone de couverture par une cellule hexagonale, par analogie à la représentation "zéro interférence" adoptée lors de la distribution des cellules des réseaux cellulaires. Il faut noter que les services $\{SE2,j_{(P2)}\}$ et $\{SE2,k_{(P3)}\}$ sont aussi ubiquitaires les uns aux autres. De plus, les services ubiquitaires $\{SE1,i_{(P1)}\}$ constituent une *Ubiquity and Provider based Virtual Service Community*, notée VSC-U&P$_{(1,1)}$, qui est associée au service *SE1* et au fournisseur P_1. De même, les services ubiquitaires $\{SE2,j_{(P2)}\}$ (respectivement $\{SE2,k_{(P3)}\}$) constituent la communauté VSC-U&P$_{(2,2)}$ (respectivement VSC-U&P$_{(2,3)}$). Les deux communautés, VSC-U&P$_{(2,2)}$ et VSC-U&P$_{(2,3)}$, sont associées toutes les deux au même type de service (*SE2*), mais à des fournisseurs différents (P_2 ou P_3).

Par conséquent, selon la Figure 8.10, chaque fournisseur représente chacune de ses VSC-U&Ps par une mosaïque de zones de couverture à tailles égales. Cependant, la taille de la zone de couverture varie d'un fournisseur à un autre, et d'un service à un autre au sein du même fournisseur. Nous constatons aussi que les zones de couvertures peuvent s'interférer dans leurs représentations géographiques, mais cela n'influence pas la QoE de l'utilisateur, puisque cette interférence n'est pas de niveau fréquentiel comme dans le cas des réseaux cellulaires. Cependant, une nouvelle problématique se pose : puisque les services n'émettent pas des signaux fréquentiels comme le font les antennes cellulaires, comment faire pour détecter le déplacement de l'utilisateur d'une zone de couverture à une autre pour initier par la suite le handover sémantique ? Pour répondre à cette problématique, nous proposons de détecter la sortie de l'utilisateur d'une zone de couverture en se basant sur sa localisation géographique fournie par son terminal et en utilisant un service de gestion déployé dans chacune des plates-formes de services.

Ainsi, chaque plate-forme effectue, selon une approche *make-before-break*, le handover sémantique qui correspond à ses services tout en restant transparent vis-à-vis de l'utilisateur.

Suite à nos études, nous distinguons deux types de handover sémantique :

- Handover sémantique intra-fournisseur : il s'applique quand l'utilisateur se déplace entre deux zones de couvertures qui correspondent à deux services ubiquitaires appartenant à un même fournisseur. Dans ce cas, le fournisseur se sert de la VSC-U&P qu'il a créée pour ses services. Par exemple, l'utilisateur se déplace de la zone de couverture d'un service $\{SE1,1_{(P1)}\}$ pré-provisionné dans le VPSN, vers la zone de couverture d'un service ubiquitaire $\{SE1,2_{(P1)}\}$ qui est fourni par le même fournisseur P_1. Dans ce cas, VSC-U&P$_{(1,1)}$ contenant ces deux services est utilisée comme support pour ce handover intra-fournisseur.
- Handover sémantique inter-fournisseurs : il s'applique quand l'utilisateur se déplace entre deux zones de couvertures qui correspondent à deux services ubiquitaires appartenant à deux fournisseurs différents. Dans ce cas, le fournisseur, auquel appartient le service pré-provisionné dans le VPSN, se sert de l'Inner VSC-U&L qu'il a créée pour son service. Par exemple, l'utilisateur se déplace de la zone de couverture d'un service $\{SE2,1_{(P2)}\}$, pré-provisionné dans le VPSN et fourni par un fournisseur P_2, vers la zone de couverture d'un service ubiquitaire $\{SE2,2_{(P3)}\}$ qui est fourni par un autre fournisseur P_3. Dans ce cas, le fournisseur P_2 se sert de l'Inner VSC-U&L$_{(2,1)}$ associée à son service $\{SE2,1_{(P2)}\}$ afin de trouver le possible remplaçant ubiquitaire $\{SE2,2_{(P3)}\}$.

Dans la sous-section suivante, nous expliquons plus en détails la procédure de handover sémantique (intra-fournisseur et inter-fournisseurs), en introduisant l'aspect fonctionnel de cette nouvelle approche.

8.2.3 Aspect fonctionnel du Handover Sémantique

Dans cette partie, nous présentons l'aspect fonctionnel de notre proposition de handover sémantique. Nous identifions les différents acteurs qui jouent les trois rôles évènementiels (initiateur, décideur et exécuteur) et nous détaillons leurs processus fonctionnels et leurs interactions.

8.2.3.1 Initiateur du handover sémantique

Le rôle d'initiateur du handover sémantique consiste, comme le montre la Figure 8.11, à effectuer la séquence suivante : au début, détecter l'évènement de mobilité spatiale, ensuite, détecter le handover sémantique et enfin, initier le processus de gestion par handover sémantique.

Afin de détecter la mobilité spatiale d'un utilisateur donné, nous intégrons dans son terminal une base de connaissances dénommée *Infosphère* [57], par analogie à l'*Infoware* qui est associé à la plate-forme de services. Comme son nom l'indique, l'*Infosphère* est une sphère d'informations qui s'attache toujours à l'utilisateur et qui l'aide à avoir la bonne décision au bon moment. Elle contient les caractéristiques du terminal (dans le *Terminal Profile*), les préférences de l'utilisateur (dans le *User Profile*), la session de

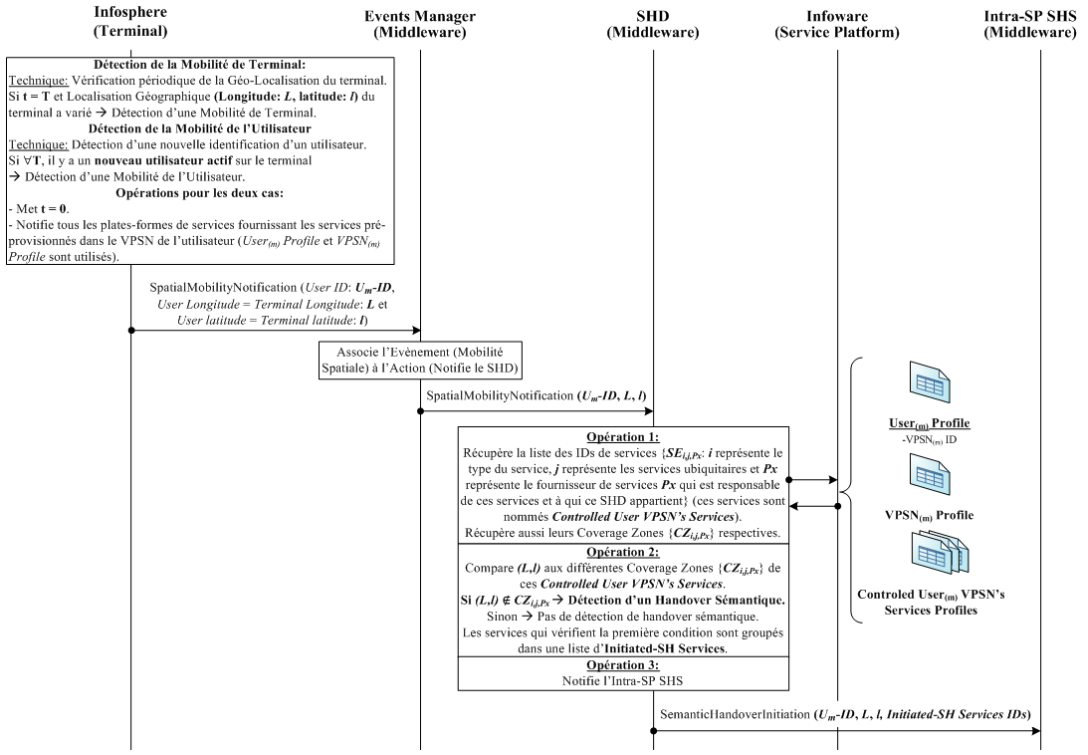

Fig. 8.11: Diagramme de séquence pour l'Initiateur du Handover Sémantique.

services pré-provisionnée (dans le *VPSN Profile*) ainsi que d'autres informations temps réel concernant les ressources ambiantes (dans l'*Active Profile*). En effet, selon l'approche évènementielle, l'*Infosphère* peut détecter deux types d'évènements :

- L'évènement de mobilité du terminal : il a lieu quand l'utilisateur se déplace tout en utilisant le même terminal. Afin de détecter cet évènement, l'*Infosphère* vérifie périodiquement, dans le *Terminal Profile*, la localisation géographique (Longitude : L, latitude : l) du terminal. Si après une période T, la localisation géographique du terminal a changé, l'*Infosphère* considère la détection d'une mobilité de terminal.
- L'évènement de mobilité de l'utilisateur : il a lieu quand l'utilisateur passe d'un terminal à un autre. Afin de détecter cet évènement, l'*Infosphère* n'a pas besoin de procéder périodiquement. En effet, si un nouvel utilisateur actif s'identifie sur un terminal donné à n'importe quel instant, l'*Infosphère* considère la détection d'une mobilité de l'utilisateur.

Ensuite, lors de la détection d'un de ces deux évènements de mobilité spatiale, l'*Infosphère* effectue la séquence suivante : premièrement, il remet le temporisateur à zéro ; deuxièmement, il récupère du *User Profile* l'identifiant du VPSN associé à l'utilisateur ; troisièmement, il récupère du *VPSN Profile* la liste des identifiants et des adresses logiques des services sélectionnés dans ce VPSN ; et finalement, il notifie toutes

les plates-formes fournissant ces services, en envoyant une notification, *Spatial Mobility Notification*, dans laquelle il précise l'identifiant de l'utilisateur ainsi que sa localisation géographique. Puisque nous adoptons une approche trans-organisationnelle, les services pré-provisionnés dans le VPSN peuvent appartenir à différentes plates-formes et à différents fournisseurs. Par conséquent, chaque plate-forme contrôle le handover sémantique qui concerne ses propres services sélectionnés dans le VPSN.

D'après notre approche évènementielle, c'est l'*Events Manager*, dans chacune des plates-formes concernées, qui va recevoir les notifications envoyées par l'*Infosphère*. Ainsi, il associe cet évènement à une action de notification du service de gestion qui est abonné à ce type d'évènement. Dans notre cas, nous proposons un service de gestion dénommé *Semantic Handover Detector* (*SHD*) qui a souscrit à l'évènement de mobilité spatiale. Ainsi, chaque *Events Manager* envoie une notification, *Spatial Mobility Notification*, vers le service *SHD* de sa plate-forme.

Comme son nom l'indique, le *SHD* détecte s'il y a un handover sémantique ou pas. Si oui, il initie le processus de gestion à base du handover sémantique en notifiant le décideur. Comme le montre la Figure 8.11, le processus fonctionnel du service *SHD* correspond à la séquence suivante d'opérations :

- Opération 1 : Puisque chaque plate-forme est responsable d'exercer le handover sémantique sur ses propres services, le *SHD* récupère de son *Infoware* la liste des identifiants des services qui sont sélectionnés par le VPSN et qui appartiennent à sa plate-forme. Ainsi, ces services sont nommés *Controlled User VPSN's Services*, et sont représentés comme suit $\{SE_{i,j,Px}$: i représente le type de service, j différencie les services ubiquitaires de même type, et P_x représente le fournisseur qui est responsable de cette plate-forme contenant ces services $SE_{i,j}\}$. Ensuite, pour que le *SHD* détecte s'il y a un handover sémantique ou non pour un de ses *Controlled User VPSN's Services*, il a besoin de savoir si l'utilisateur est sorti ou non de la zone de couverture de ce service. Pour cette raison, dans cette même opération, le *SHD* récupère aussi de l'*Infoware*, pour chacun de ses *Controlled User VPSN's Services*, la zone de couverture $CZ_{i,j,Px}$ qui lui correspond.
- Opération 2 : Le *SHD* compare la localisation géographique (L,l) du terminal à la liste des zones de couvertures récupérées $\{CZ_{i,j,Px}\}$. Pour chacun des *Controlled User VPSN's Services*, si (L,l) n'appartient pas à sa zone de couverture, le *SHD* considère une détection d'un handover sémantique pour ce service. Sinon, le *SHD* considère qu'il n'y pas de handover sémantique pour ce service. Par conséquent, les services qui vérifient la première condition sont ajoutés à une liste de services, dénommée *Initiated-SH Services* (c.à.d. la liste des services pour lesquels, le *SHD* va initier un handover sémantique).
- Opération 3 : Enfin, le *SHD* initie le handover sémantique en notifiant le décideur de sa propre plate-forme. Pour cela, il envoie une notification, *Semantic Handover Initiation*, au décideur tout en précisant l'identifiant de l'utilisateur, sa localisation géographique ainsi que la liste des *Initiated-SH Services*.

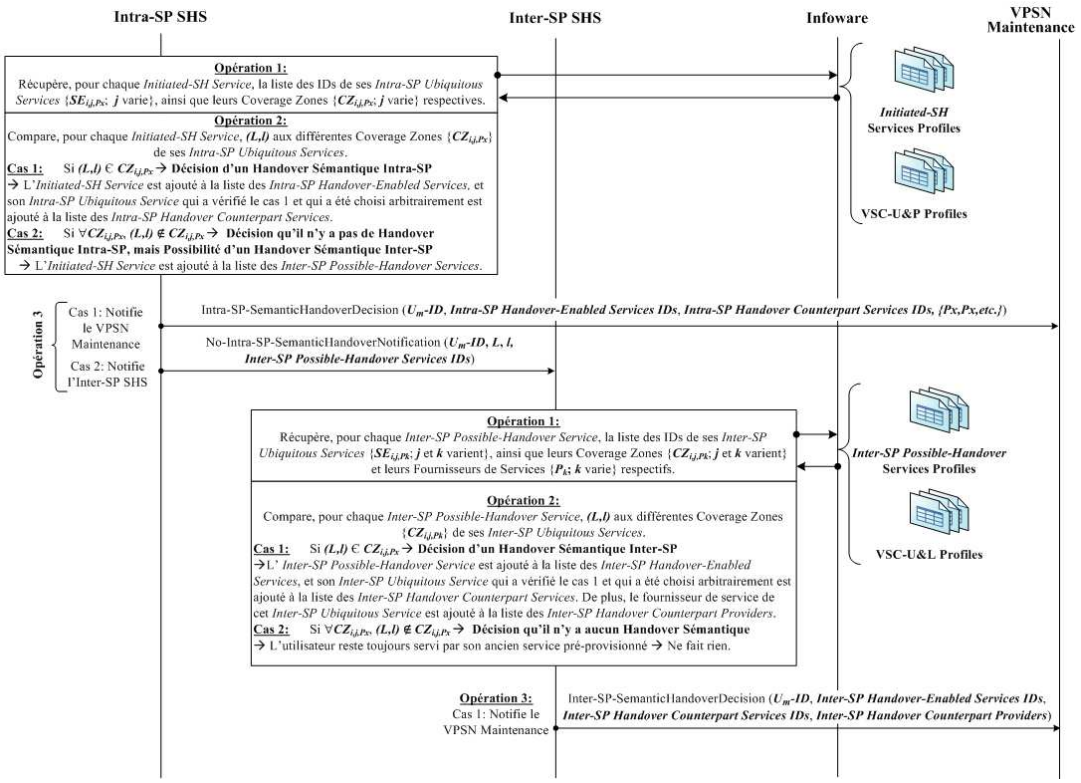

Fig. 8.12: Diagramme de séquence pour le Décideur du Handover Sémantique.

8.2.3.2 Décideur du handover sémantique

Le rôle de décideur du handover sémantique consiste à prendre les décisions de remplacement des *Initiated-SH Services* par d'autres services qui sont ubiquitaires et qui appartiennent à la zone ambiante de l'utilisateur. Comme le montre la Figure 8.12, le décideur différencie deux types de décisions possibles pour le handover sémantique :

- Décision d'un handover sémantique intra-fournisseur : elle est prise quand, pour un *Initiated-SH Service* donné, il y a au moins un service ubiquitaire qui appartient à son fournisseur et qui couvre, selon sa zone de couverture, la nouvelle localisation géographique du terminal. Il faut noter que ce service ubiquitaire peut appartenir à différentes plates-formes de ce même fournisseur. Afin de supporter cette décision de handover sémantique intra-fournisseur, le décideur utilise pour chaque *Initiated-SH Service* la communauté virtuelle VSC-U&P qui lui correspond. Cette communauté déjà créée contient l'ensemble des *Intra-SP Ubiquitous Services* qui sont ubiquitaires à cet *Initiated-SH Service* et qui sont aussi offerts par son propre fournisseur.
- Décision d'un handover sémantique inter-fournisseurs : elle est prise quand, pour un *Initiated-SH Service* donné, deux conditions sont vérifiées : la première consiste que le décideur n'a trouvé aucun *Intra-SP Ubiquitous Service* qui couvre la nouvelle localisation géographique du terminal ; et la seconde consiste que le décideur

a trouvé au moins un service ubiquitaire qui appartient à un autre fournisseur et qui couvre, selon sa zone de couverture, la nouvelle localisation géographique du terminal. Afin de supporter cette décision de handover sémantique inter-fournisseurs, le décideur utilise pour chaque *Initiated-SH Service* la communauté virtuelle Inner VSC-U&L qui lui correspond. L'usage de cette communauté déjà créée est avantageux, puisqu'elle contient l'ensemble des *Inter-SP Ubiquitous Services* qui sont multi-fournisseurs, ubiquitaires à cet *Initiated-SH Service*, et qui appartiennent à sa zone d'entourage. Ainsi, quand le terminal sort de la zone de couverture de l'*Initiated-SH Service*, c'est plus probable de trouver, parmi ces *Inter-SP Ubiquitous Service* qui se trouvent dans son entourage, un service dont la zone de couverture contient la nouvelle localisation du terminal.

Normalement, un fournisseur de services préfère un handover sémantique intra-fournisseur, car il lui permet d'éviter de passer par un autre fournisseur et de payer à ce dernier l'utilisation d'un de ses services ubiquitaires. Par conséquent, dans notre proposition, nous favorisons le handover sémantique intra-fournisseur par rapport au handover sémantique inter-fournisseurs. Pour ce faire, nous proposons un nouveau service de gestion, dénommé *Intra Service Provider Semantic Handover Service* (*Intra-SP SHS*), qui intervient en premier afin de décider s'il y a un handover sémantique intra-fournisseur ou non. Par conséquent, c'est l'*Intra-SP SHS* qui reçoit l'évènement, *Semantic Handover Initiation*, qui est envoyé par l'initiateur. Comme le montre la Figure 8.12, le processus fonctionnel du service *Intra-SP SHS* correspond à la séquence suivante d'opérations :

- Opération 1 : Pour chaque *Initiated-SH Service*, l'*Intra-SP SHS* récupère de l'*Infoware* la liste des identifiants des *Intra-SP Ubiquitous Services*, notés $\{SE_{i,j,Px}$: seulement j varie$\}$, ainsi que leurs zones de couvertures, notées $\{CZ_{i,j,Px}$: seulement j varie$\}$. Dans ce but, les profils de l'*Initiated-SH Service* et de sa VSC-U&P correspondante sont utilisés.
- Opération 2 : Ensuite, l'*Intra-SP SHS* compare, pour chaque *Initiated-SH Service*, la localisation géographique (L,l) du terminal aux différentes zones de couvertures $\{CZ_{i,j,Px}\}$ des *Intra-SP Ubiquitous Services*. Par conséquent, deux cas sont possibles. Le premier cas aura lieu quand (L,l) appartient au moins à une zone de couverture. Dans ce cas, l'*Intra-SP SHS* considère une décision d'un handover sémantique intra-fournisseur pour cet *Initiated-SH Service*. Par conséquent, il choisit arbitrairement un des services dont la zone de couverture contient (L,l), et il l'ajoute à la liste des *Intra-SP Handover Counterpart Services* qui correspondent à la liste des remplaçants ubiquitaires intra-fournisseur. De plus, il ajoute l' *Initiated-SH Service* à la liste des *Intra-SP Handover-Enabled Services* qui vont subir le remplacement intra-fournisseur. Au contraire, le deuxième cas possible aura lieu quand, pour un *Initiated-SH Service* donné, (L,l) n'appartient à aucune des zones de couverture des *Intra-SP Ubiquitous Services*. Par conséquent, il n'y a pas de handover sémantique intra-fournisseur. Cependant, une décision d'un handover sémantique inter-fournisseurs est toujours possible. Pour cela, l'ensemble des *Initiated-SH Services* qui vérifient le deuxième cas sont ajoutés à la liste des

Inter-SP Possible-Handover Services qui sont susceptibles de subir un handover sémantique inter-fournisseurs.

- Opération 3 : Pour le premier cas, l'*Intra-SP SHS* notifie l'exécuteur du handover sémantique en envoyant l'évènement *Intra-SP Semantic Handover Decision*. Ce dernier contient l'identifiant de l'utilisateur, la liste des *Intra-SP Handover-Enabled Services*, la liste des *Intra-SP Handover Counterpart Services* relatifs, ainsi que la liste de leurs *Intra-SP Handover Counterpart Providers*. Ces derniers correspondent à une liste de P_x, puisque nous sommes dans le cas d'un handover sémantique intra-fournisseur, ce qui implique que nous avons le même fournisseur P_x. Pour le deuxième cas, une décision d'un handover sémantique intra-fournisseur n'est pas prise. Par contre, une décision d'un handover sémantique inter-fournisseurs est toujours possible. Pour cela, l'*Intra-SP SHS* notifie un autre acteur, dénommé *Inter Service Providers Semantic Handover Service* (*Inter-SP SHS*), en envoyant la notification, *No Intra-SP Semantic Handover Notification*. Cette dernière contient l'identifiant de l'utilisateur, sa localisation géographique, ainsi que la liste des *Inter-SP Possible-Handover Services*.

Après avoir expliqué le processus fonctionnel exécuté par l'*Intra-SP SHS*, nous prenons le cas où l'*Intra-SP SHS* prend la décision qu'il n'y a pas de handover sémantique intra-fournisseur, et il envoie la notification, *No Intra-SP Semantic Handover Notification*, vers l'*Inter-SP SHS*. Ce dernier il a pour rôle de décider s'il y a un handover sémantique inter-fournisseurs ou non. Comme le montre la Figure 8.12, le processus fonctionnel du service *Inter-SP SHS* correspond à la séquence suivante d'opérations :

- Opération 1 : Pour chaque *Inter-SP Possible-Handover Service*, l'*Inter-SP SHS* récupère de l'*Infoware* la liste des identifiants des *Inter-SP Ubiquitous Services*, notés $\{SE_{i,j,Pk} : j$ et k varient, avec $P_k \neq P_x\}$, ainsi que leurs zones de couvertures, notées $\{CZ_{i,j,Pk} : j$ et k varient, avec $P_k \neq P_x\}$. Dans ce but, les profils de l'*Inter-SP Possible-Handover Service* et de l'Inner VSC-U&L correspondante sont utilisés.
- Opération 2 : Ensuite, l'*Inter-SP SHS* compare, pour chaque *Inter-SP Possible-Handover Service*, la localisation géographique (L,l) du terminal aux différentes zones de couvertures $\{CZ_{i,j,Pk}\}$ des *Inter-SP Ubiquitous Services*. Par conséquent, deux cas sont possibles. Le premier cas aura lieu quand (L,l) appartient au moins à une zone de couverture. Dans ce cas, l'*Inter-SP SHS* considère une décision d'un handover sémantique inter-fournisseurs pour cet *Inter-SP Possible-Handover Service*. Par conséquent, il choisit arbitrairement un des services dont la zone de couverture contient (L,l), et il l'ajoute à la liste des *Inter-SP Handover Counterpart Services* qui correspondent à la liste des remplaçants ubiquitaires inter-fournisseurs. De plus, il ajoute l'identifiant du fournisseur, qui offre le service choisi, à la liste des *Inter-SP Handover Counterpart Providers*. Enfin, il ajoute l'*Inter-SP Possible-Handover Service* à la liste des *Inter-SP Handover-Enabled Services* qui vont subir le remplacement inter-fournisseurs. Au contraire, le deuxième cas possible aura lieu quand, pour un *Inter-SP Possible-Handover Service* donné, (L,l) n'appartient à aucune des zones de couverture des *Inter-*

SP Ubiquitous Services. Par conséquent, il n'y a pas de handover sémantique inter-fournisseurs. Par suite, ce service ne subit ni un handover sémantique intra-fournisseur, ni un handover sémantique inter-fournisseurs. Ainsi, ce service est gardé dans le VPSN de l'utilisateur.

– Opération 3 : Pour le premier cas où nous avons une décision d'un handover sémantique inter-fournisseurs, l'*Inter-SP SHS* notifie l'exécuteur du handover sémantique en envoyant l'évènement *Inter-SP Semantic Handover Decision*. Ce dernier contient l'identifiant de l'utilisateur, la liste des *Inter-SP Handover-Enabled Services*, la liste des *Inter-SP Handover Counterpart Services* relatifs, ainsi que la liste de leurs *Inter-SP Handover Counterpart Providers*. Ces derniers correspondent à une liste $\{P_k$: k varie, avec $P_k \neq P_x\}$.

8.2.3.3 Exécuteur du handover sémantique

Le rôle d'exécuteur du handover sémantique est joué, comme le montre la Figure 8.13, par le service *VPSN Maintenance* qui interagit avec l'*Infoware* afin d'exécuter la décision envoyée par le décideur. Comme son nom l'indique, le *VPSN Maintenance* maintient le VPSN de l'utilisateur tout au long de sa mobilité spatiale et temporelle, en remplaçant dans le *VPSN Profile* les services pré-provisionnés par d'autres services ubiquitaires appartenant au nouveau contexte ambiant de l'utilisateur. Au niveau fonctionnel, le *VPSN Maintenance* reçoit deux types de décisions : le premier consiste

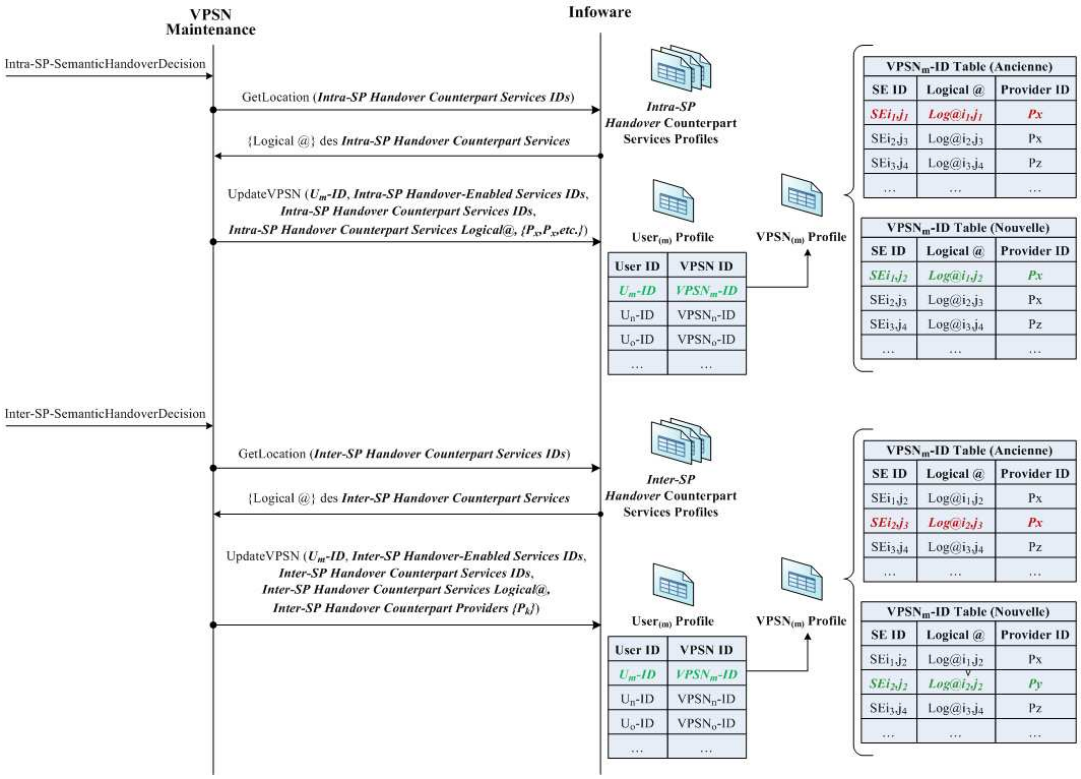

Fig. 8.13: Diagramme de séquence pour l'Exécuteur du Handover Sémantique.

à recevoir la notification, *Intra-SP Semantic Handover Decision*, qui est envoyée par l'*Intra-SP SHS*, et le deuxième consiste à recevoir la notification, *Inter-SP Semantic Handover Decision*, qui est envoyée par l'*Inter-SP SHS*. Ces deux types de notifications contiennent des informations différentes, mais le *VPSN Maintenance* les traite de la même façon. En effet, le processus fonctionnel du service *VPSN Maintenance* correspond à la séquence suivante d'opérations :

- Opération 1 : Le *VPSN Maintenance* récupère de l'*Infoware* la liste des adresses logiques qui correspondent à la liste des *Intra-SP Handover Counterpart Services* (respectivement *Inter-SP Handover Counterpart Services*). Dans ce but, les profils de ces services sont utilisés.
- Opération 2 : Le *VPSN Maintenance* adapte le VPSN de l'utilisateur en envoyant un évènement, *Update VPSN*, à l'*Infoware*. Dans le cas d'un handover sémantique intra-fournisseur, l'*Update VPSN* contient l'identifiant de l'utilisateur, la liste d'identifiants des *Intra-SP Handover-Enabled Services*, ainsi que la liste des identifiants, des adresses logiques et des *Providers IDs* correspondant aux *Intra-SP Handover Counterpart Services*. Dans le cas d'un handover sémantique inter-fournisseurs, l'*Update VPSN* contient l'identifiant de l'utilisateur, la liste d'identifiants des *Inter-SP Handover-Enabled Services*, ainsi que la liste des identifiants, des adresses logiques et des *Providers IDs* correspondant aux *Inter-SP Handover Counterpart Services*.

Enfin, l'*Infoware* reçoit l'évènement *Update VPSN*. Il utilise l'identifiant de l'utilisateur afin de récupérer du *User Profile* l'identifiant de son VPSN. Ensuite, il adapte le *VPSN Profile* en remplaçant les identifiants, les adresses logiques et les *Provider IDs* relatifs aux *Intra-SP Handover-Enabled Services* (respectivement *Inter-SP Handover-Enabled Services*) par ceux des *Intra-SP Handover Counterpart Services* (respectivement *Inter-SP Handover Counterpart Services*). En effet, la Figure 8.13, montre comment le *VPSN Profile*, qui est représenté sous forme d'une table relationnelle, est adapté. Par exemple, suite à l'exécution d'une décision d'un handover sémantique intra-fournisseur, le triplet ($SE_{i1,j1}$; $Log@_{i1,j1}$; P_x) d'un service donné est remplacé par le triplet ($SE_{i1,j2}$; $Log@_{i1,j2}$; P_x) correspondant à un autre service ubiquitaire qui appartient au même fournisseur P_x. De plus, suite à l'exécution d'une décision d'un handover sémantique inter-fournisseurs, le triplet ($SE_{i2,j1}$; $Log@_{i2,j1}$; P_x) d'un service donné est remplacé par le triplet ($SE_{i2,j2}$; $Log@_{i2,j2}$; P_y) correspondant à un autre service ubiquitaire qui appartient à un autre fournisseur P_y.

Puisque dans notre étude nous adoptons une approche trans-organisationnelle à base d'une gestion distribuée, le fragment d'*Infoware* qui a effectué l'adaptation du VPSN doit notifier les autres fragments d'*Infoware* qui sont concernés par ces modifications. Par conséquent, des requêtes de mise à jour sont envoyées vers l'*Infosphère* associée au terminal de l'utilisateur, vers les fragments d'*Infoware* associés aux *Intra-SP Handover Counterpart Services* et aux *Inter-SP Handover Counterpart Services*, et enfin vers les fragments d'*Infoware* associés aux autres services qui sont pré-provisionnés par ce VPSN, mais qui n'ont pas subi de modifications.

8.2.4 Mise en œuvre du Handover Sémantique

Afin de mettre en exergue l'approche évènementielle et distribuée que nous proposons pour la gestion de mobilité à l'aide du handover sémantique, nous présentons dans cette sous-section une mise en œuvre organisationnelle d'un handover sémantique intra-fournisseur (voir Figure 8.15) et d'un handover sémantique inter-fournisseurs (voir Figure 8.16), lors d'une mobilité de terminal (l'utilisateur se déplace tout en utilisant le même terminal). Dans la suite, l'approche évènementielle est représentée, comme dans le cas des communautés virtuelles, par un ensemble d'évènements classifiés sous deux types :

- $Event_{(c)}$: il représente un *Computing Event* échangé entre un service et l'*Infoware*, ou entre deux services.
- $Event_{(i)}$: il représente un *Informational Event* échangé entre deux fragments d'*Infoware* distribués.

Pour mettre en œuvre la solution de handover sémantique, nous reprenons l'exemple des trois fournisseurs de services (P_1, P_2, P_3) qui ont distribué leurs services ubiquitaires respectifs ($\{SE1,i_{(P1)}\}$, $\{SE2,j_{(P2)}\}$, $\{SE2,k_{(P3)}\}$) sur différentes zones de couvertures, comme le montre la Figure 8.14. Les services ubiquitaires $\{SE1,i_{(P1)}\}$ constituent la communauté VSC-U&P$_{(1,1)}$. De même, les services ubiquitaires $\{SE2,j_{(P2)}\}$ et $\{SE2,k_{(P3)}\}$ constituent les communautés VSC-U&P$_{(2,2)}$ et VSC-U&P$_{(2,3)}$ respectivement. Ainsi, chaque fournisseur représente sa VSC-U&P par une mosaïque de zones de couverture, et il associe une zone de couverture à chacun de ses services ubiquitaires.

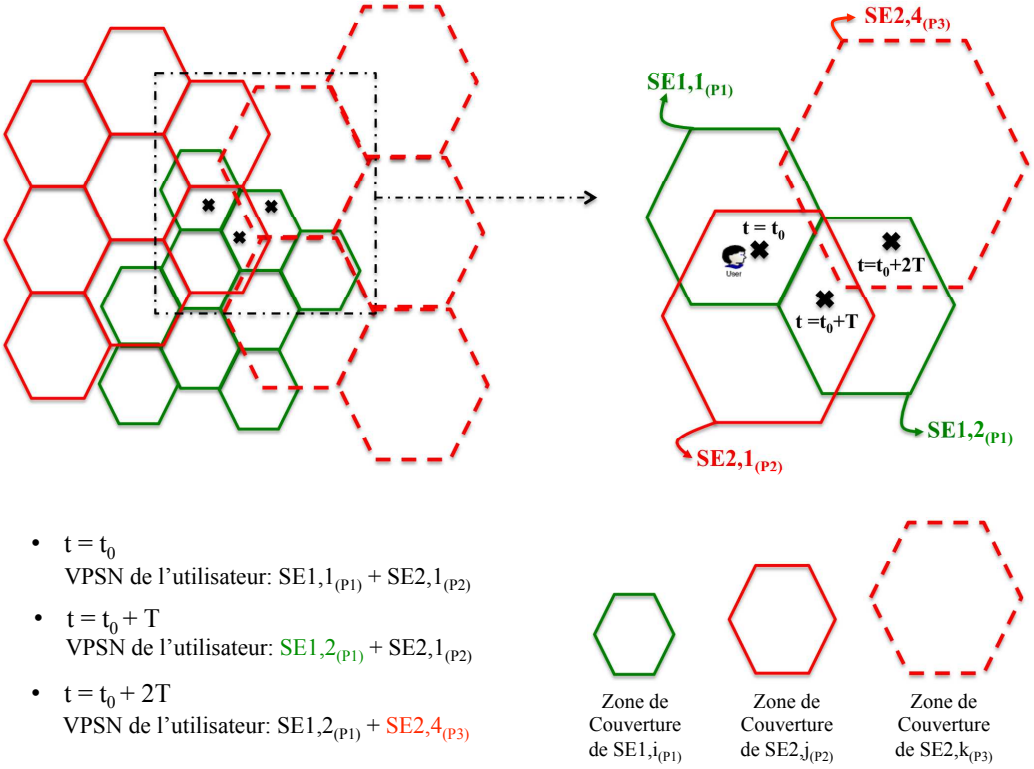

Fig. 8.14: Cas d'usage pour la mise en œuvre du Handover Sémantique.

Il faut noter que les services $\{SE2,j_{(P2)}\}$ et $\{SE2,k_{(P3)}\}$ sont aussi ubiquitaires les uns aux autres.

Selon la Figure 8.14, l'utilisateur possède initialement un VPSN pré-provisionné contenant les éléments de services suivants : $SE1,1_{(P1)}$ et $SE2,1_{(P2)}$. Donc, à $t{=}t_0$, la localisation géographique (L_0,l_0) de l'utilisateur appartient aux zones de couvertures de ces deux services. Ensuite, nous considérons que l'utilisateur se déplace. Puisque la vérification de la localisation géographique de l'utilisateur se fait d'une manière périodique, nous considérons qu'après une période T, il passe de (L_0,l_0) à une nouvelle localisation (L_1,l_1).

La Figure 8.15 montre comment cette mobilité du terminal est détectée au sein du terminal et comment elle va impacter le VPSN de l'utilisateur. En effet, à $t{=}t_0{+}T$, dans l'*Infosphère* associée au terminal, le champs "*geolocation*" du *Terminal Profile* a changé de (L_0,l_0) à (L_1,l_1). Par conséquent, l'*Infosphère* exécute la séquence suivante : elle détecte la mobilité du terminal, récupère l'identifiant du VPSN à partir du *User Profile*, récupère les adresses logiques relatifs aux services du VPSN à partir du *VPSN Profile*, et enfin, envoie un évènement noté $Event_{(c)}$ *1* à toutes ces adresses logiques, tout en précisant l'identifiant et la nouvelle localisation géographique de l'utilisateur. Dans ce cas, les adresses logiques correspondent aux plates-formes $P1_{(1)}$ et $P2_{(1)}$ qui fournissent respectivement les deux services $SE1,1_{(P1)}$ et $SE2,1_{(P2)}$ sélectionnés dans le VPSN. Ainsi, chacune de ces plates-formes lance son propre processus de handover

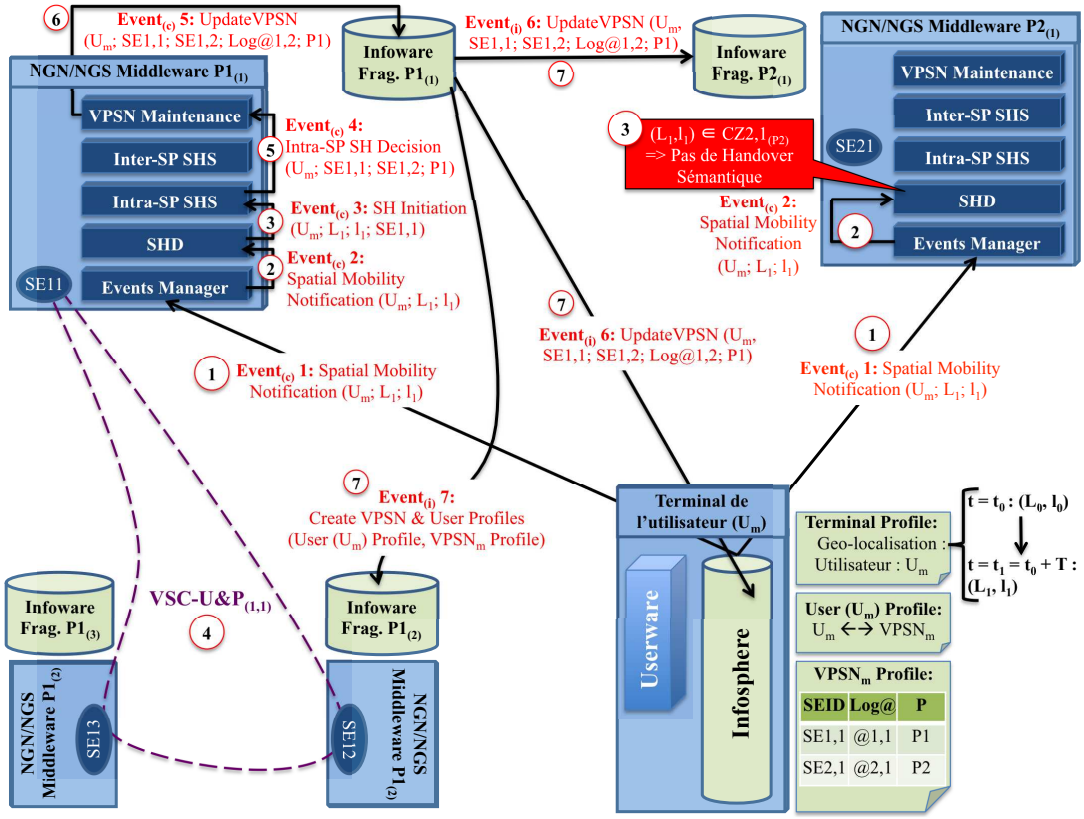

Fig. 8.15: Mise en œuvre d'un Handover Sémantique Intra Fournisseur.

sémantique, d'une manière parallèle, distribuée et indépendante.

$Event_{(c)}$ 1 : Spatial Mobility Notification (U_m ; L_1 ; l_1)

Comme le montre la Figure 8.15, dans les deux plates-formes, un *Events Manager* reçoit l'*$Event_{(c)}$ 1* et l'associe au service de gestion qui lui est abonné. Ainsi, chaque *Events Manager* notifie le *SHD* de sa plate-forme, en envoyant un évènement *$Event_{(c)}$ 2*.

$Event_{(c)}$ 2 : Spatial Mobility Notification (U_m ; L_1 ; l_1)

Ainsi, chaque *SHD* est responsable de détecter s'il y aura une initiation d'un handover sémantique ou non. En effet, le *SHD* de la plate-forme *$P2_{(1)}$* trouve que (L_1,l_1) $\in$ *$CZ2,1_{(P2)}$*, ce qui implique que l'utilisateur se trouve toujours dans la zone de couverture de *$SE2,1_{(P2)}$* (ce qui apparaît aussi dans la Figure 8.14 à t=t_0+T). Par conséquent, le *SHD* de la plate-forme *$P2_{(1)}$* décide de ne pas initier un handover sémantique pour le service *$SE2,1_{(P2)}$* et le garde ainsi dans le VPSN de l'utilisateur. Au contraire, le *SHD* de la plate-forme *$P1_{(1)}$* trouve que (L_1,l_1) $\notin$ *$CZ1,1_{(P1)}$*, ce qui implique que l'utilisateur se trouve en dehors de la zone de couverture de *$SE1,1_{(P1)}$* (ce qui apparaît aussi dans la Figure 8.14 à t=t_0+T). Par conséquent, le *SHD* de la plate-forme *$P1_{(1)}$* décide d'initier un handover sémantique pour le service *$SE1,1_{(P1)}$*. Pour cela, il envoie un évènement, *$Event_{(c)}$ 3*, à l'*Intra-SP SHS* qui joue le rôle de décideur.

$Event_{(c)}$ 3 : Semantic Handover Initiation (U_m ; L_1 ; l_1 ; $SE1,1_{(P1)}$)

Ensuite, l'*Intra-SP SHS* compare (L_1,l_1) à l'ensemble des zones de couvertures relatives aux services sélectionnés dans la VSC-U&P$_{(1,1)}$. En effet, ces services sont ubiquitaires à *$SE1,1_{(P1)}$* qui va subir le handover sémantique, et appartiennent au même fournisseur P_1. Suite à cette comparaison, l'*Intra-SP SHS* trouve que (L_1,l_1) $\in$ *$CZ1,2_{(P1)}$*, ce qui implique que l'utilisateur se trouve dans la zone de couverture du service *$SE1,2_{(P1)}$* (ce qui apparaît aussi dans la Figure 8.14 à t=t_0+T). Par conséquent, l'*Intra-SP SHS* prend une décision d'un handover sémantique intra-fournisseur. Ainsi, il envoie un évènement, *$Event_{(c)}$ 4*, au service *VPSN Maintenance* qui joue le rôle d'exécuteur. Dans cet évènement, l'*Intra-SP SHS* précise l'identifiant de l'utilisateur, l'identifiant de l'*Intra-SP Handover-Enabled Service* (*$SE1,1_{(P1)}$*) qui va subir le handover, l'identifiant de l'*Intra-SP Handover Counterpart Service* (*$SE1,2_{(P1)}$*) qui va le remplacer dans le VPSN, ainsi que l'identifiant du fournisseur du service *$SE1,2_{(P1)}$*.

$Event_{(c)}$ 4 : Intra-SP Semantic Handover Decision (U_m ; $SE1,1_{(P1)}$; $SE1,2_{(P1)}$; P_1)

Ensuite, le *VPSN Maintenance* exécute la décision en envoyant un évènement, *$Event_{(c)}$ 5*, à l'*Infoware* pour que ce dernier mette à jour le *VPSN Profile*. Dans cet évènement, le *VPSN Maintenance* précise l'identifiant de l'utilisateur, l'identifiant de *$SE1,1_{(P1)}$* à remplacer, l'identifiant de *$SE1,2_{(P1)}$* remplaçant, l'adresse logique de *$SE1,2_{(P1)}$* ainsi que l'identifiant de son fournisseur. Par conséquent, le triplet

($SE1,1_{(P1)}$; $Log@1,1$; P_1) est remplacé par le triplet ($SE1,2_{(P1)}$; $Log@1,2$; P_1).

$Event_{(c)}$ 5 : Update VPSN (U_m ; $SE1,1_{(P1)}$; $SE1,2_{(P1)}$; $Log@1,2$; P_1)

Enfin, cet *Infoware* associé à la plate-forme $P1_{(1)}$ envoie un évènement, $Event_{(i)}$ *6*, à l'*Infosphère* associée au terminal et au fragment d'*Infoware* associé à la plate-forme $P2_{(1)}$ qui héberge l'autre service du VPSN ($SE2,1_{(P2)}$) et qui possède aussi le profil du VPSN adapté. Ainsi, ce fragment d'*Infoware* et cet *Infosphère* adaptent eux aussi le *VPSN Profile* correspondant à cet utilisateur mobile. En outre, l'*Infoware* associé à la plate-forme $P1_{(1)}$ envoie un autre type d'évènement, $Event_{(i)}$ *7*, au fragment d'*Infoware* associé à la plate-forme $P1_{(2)}$ qui héberge le service $SE1,2_{(P1)}$ utilisé comme remplaçant. Par conséquent, ce fragment crée un nouveau *VPSN Profile* et un nouveau *User Profile* correspondant à l'utilisateur mobile. Par la suite, ce fragment d'*Infoware* associé à la plate-forme $P1_{(2)}$ devient responsable de la maintenance du VPSN lors d'une mobilité spatiale, au lieu du fragment d'*Infoware* associé à la plate-forme $P1_{(1)}$.

$Event_{(i)}$ 6 : Update VPSN (U_m ; $SE1,1_{(P1)}$; $SE1,2_{(P1)}$; $Log@1,2$; P_1)

$Event_{(i)}$ 7 : Create VPSN & User Profiles (U_m Profile ; $VPSN_m$ Profile)

Comme le montre la Figure 8.14 à t=t_0+T, le nouveau VPSN de l'utilisateur est constitué, suite au handover sémantique intra-fournisseur, des deux services : $SE1,2_{(P1)}$ et $SE2,1_{(P2)}$. Maintenant, nous allons prendre le cas d'un handover sémantique inter-fournisseurs. Pour ce faire, nous considérons que l'utilisateur continue à se déplacer. À t=t_0+$2T$, c.à.d. après une autre période T, l'utilisateur passe de (L_1,l_1) à (L_2,l_2).

D'après la Figure 8.16, l'*Infosphère* associée au terminal détecte cette mobilité du terminal grâce au champs "*geolocation*" du *Terminal Profile*. Par conséquent, l'*Infosphère* exécute la même séquence qui a été faite après la première période (à t=t_0+T). Ensuite, il envoie l'évènement $Event_{(c)}$ *1* aux plates-formes $P1_{(2)}$ et $P2_{(1)}$ qui fournissent respectivement les deux services $SE1,2_{(P1)}$ et $SE2,1_{(P2)}$ sélectionnés dans le VPSN. Ainsi, chacune de ces plates-formes lance son propre processus de handover sémantique, d'une manière parallèle, distribuée et indépendante.

$Event_{(c)}$ 1 : Spatial Mobility Notification (U_m ; L_2 ; l_2)

Comme le montre la Figure 8.16, pour chacune de ces deux plates-formes, l'*Events Manager* reçoit l'$Event_{(c)}$ *1* et notifie le *SHD* correspondant, en envoyant un évènement $Event_{(c)}$ *2*.

$Event_{(c)}$ 2 : Spatial Mobility Notification (U_m ; L_2 ; l_2)

Ainsi, le *SHD* de la plate-forme $P1_{(2)}$ trouve que (L_2,l_2) $\in$ $CZ1,2_{(P1)}$, ce qui implique que l'utilisateur se trouve toujours dans la zone de couverture de $SE1,2_{(P1)}$ (ce

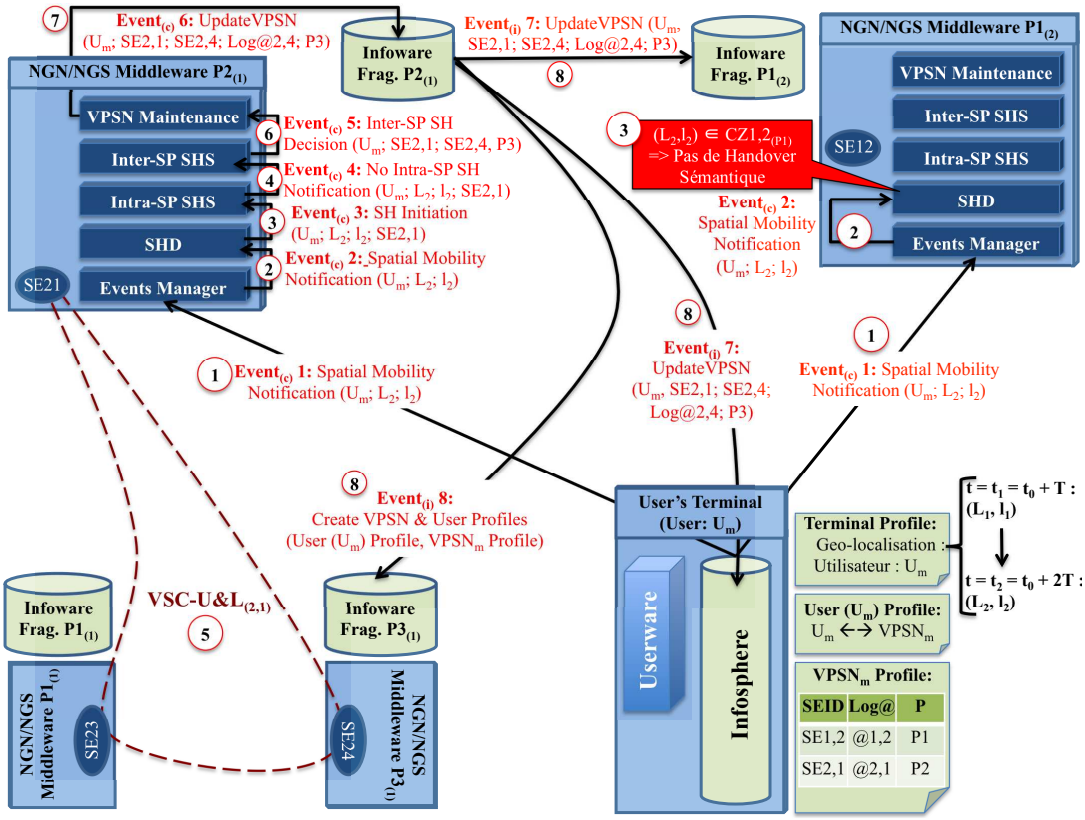

Fig. 8.16: Mise en œuvre d'un Handover Sémantique Inter Fournisseurs.

qui apparaît aussi dans la Figure 8.14 à $t{=}t_0{+}2T$). Par conséquent, le *SHD* de la plate-forme $P1_{(2)}$ décide de ne pas initier un handover sémantique pour le service $SE1,2_{(P1)}$ et le garde ainsi dans le VPSN de l'utilisateur. Au contraire, le *SHD* de la plate-forme $P2_{(1)}$ trouve que $(L_2,l_2) \notin CZ2,1_{(P2)}$, ce qui implique que l'utilisateur se trouve en dehors de la zone de couverture de $SE2,1_{(P2)}$ (ce qui apparaît aussi dans la Figure 8.14 à $t{=}t_0{+}2T$). Par conséquent, le *SHD* de la plate-forme $P2_{(1)}$ décide d'initier un handover sémantique pour le service $SE2,1_{(P2)}$. Pour cela, il envoie un évènement, $Event_{(c)}$ *3*, à l'*Intra-SP SHS* qui joue le rôle de décideur.

$Event_{(c)}$ *3 : Semantic Handover Initiation* $(U_m\ ;\ L_2\ ;\ l_2\ ;\ SE2,1_{(P2)})$

L'*Intra-SP SHS* compare (L_2,l_2) à l'ensemble des zones de couvertures relatives aux services sélectionnés dans la VSC-U&P$_{(2,2)}$. En effet, ces services sont ubiquitaires à $SE2,1_{(P2)}$ qui va subir le handover sémantique, et appartiennent au même fournisseur P_2. Suite à cette comparaison, l'*Intra-SP SHS* trouve que $(L_2,l_2) \notin \{CZ2,j_{(P2)}\}$, ce qui implique que l'utilisateur ne se trouve dans aucune des zones des couvertures associées aux services ubiquitaires du fournisseur P_2. Par conséquent, l'*Intra-SP SHS* décide que le service $SE2,1_{(P2)}$ ne vas pas subir un handover sémantique intra-fournisseur. Cependant, il est toujours possible qu'il subit un handover sémantique inter-fournisseurs. Pour cela, l'*Intra-SP SHS* envoie une notification, $Event_{(c)}$ *4*, à l'*Inter-SP SHS* pour

qu'il prenne la décision.

$Event_{(c)}$ 4 : No Intra-SP Semantic Handover Notification (U_m ; L_2 ; l_2 ; $SE2{,}1_{(P2)}$)

L'*Inter-SP SHS* compare (L_2,l_2) à l'ensemble des zones de couvertures relatives aux services sélectionnés dans la VSC-U&L$_{(2,1)}$. En effet, cette communauté est l'Inner VSC-U&L du service $SE2{,}1_{(P2)}$. Elle contient des services multi-fournisseurs, ubiquitaires à $SE2{,}1_{(P2)}$ et qui appartiennent à sa zone d'entourage. Cela permet de vérifier si dans l'entourage du service $SE2{,}1_{(P2)}$ il y a un service d'un autre fournisseur, qui soit ubiquitaire à $SE2{,}1_{(P2)}$ et dont la zone de couverture couvre bien la nouvelle localisation de l'utilisateur. Par conséquent, suite à la comparaison, l'*Inter-SP SHS* trouve que (L_2,l_2) $\in \{CZ2{,}4_{(P3)}\}$, ce qui implique que l'utilisateur se trouve dans la zone de couverture du service $SE2{,}4_{(P3)}$ (ce qui apparaît aussi dans la Figure 8.14 à t=t_0+$2T$). Par conséquent, l'*Inter-SP SHS* prend une décision d'un handover sémantique inter-fournisseurs. Ainsi, il envoie un évènement, $Event_{(c)}$ *5*, au service *VPSN Maintenance* qui joue le rôle d'exécuteur. Dans cet évènement, l'*Inter-SP SHS* précise l'identifiant de l'utilisateur, l'identifiant de l'*Inter-SP Handover-Enabled Service* ($SE2{,}1_{(P2)}$) qui va subir le handover, l'identifiant de l'*Inter-SP Handover Counterpart Service* ($SE2{,}4_{(P3)}$) qui va le remplacer dans le VPSN, ainsi que l'identifiant du fournisseur du service $SE2{,}4_{(P3)}$.

$Event_{(c)}$ 5 : Inter-SP Semantic Handover Decision (U_m ; $SE2{,}1_{(P2)}$; $SE2{,}4_{(P3)}$; P_3)

Ensuite, le *VPSN Maintenance* exécute la décision en envoyant un évènement, $Event_{(c)}$ *6*, à l'*Infoware* pour que ce dernier mette à jour le *VPSN Profile*. Dans cet évènement, le *VPSN Maintenance* précise l'identifiant de l'utilisateur, l'identifiant de $SE2{,}1_{(P2)}$ à remplacer, l'identifiant de $SE2{,}4_{(P3)}$ remplaçant, l'adresse logique de $SE2{,}4_{(P3)}$ ainsi que l'identifiant de son fournisseur. Par conséquent, le triplet ($SE2{,}1_{(P2)}$; $Log@2{,}1$; P_2) est remplacé par le triplet ($SE2{,}4_{(P3)}$; $Log@2{,}4$; P_3).

$Event_{(c)}$ 6 : Update VPSN (U_m ; $SE2{,}1_{(P2)}$; $SE2{,}4_{(P3)}$; $Log@2{,}4$; P_3)

Enfin, cet *Infoware* associé à la plate-forme $P2_{(1)}$ envoie un évènement, $Event_{(i)}$ *7*, à l'*Infosphère* associée au terminal et au fragment d'*Infoware* associé à la plate-forme $P1_{(2)}$ qui héberge l'autre service du VPSN ($SE1{,}2_{(P1)}$) et qui possède aussi le profil du VPSN adapté. Ainsi ce fragment d'*Infoware* et cet *Infosphère* adaptent eux aussi le *VPSN Profile* correspondant à cet utilisateur mobile. En outre, l'*Infoware* associé à la plate-forme $P2_{(1)}$ envoie un autre type d'évènement, $Event_{(i)}$ *8*, au fragment d'*Infoware* associé à la plate-forme $P3_{(1)}$ qui héberge le service $SE2{,}4_{(P3)}$ utilisé comme remplaçant. Par conséquent, ce fragment crée un nouveau *VPSN Profile* et un nouveau *User Profile* correspondant à l'utilisateur mobile. Par la suite, ce fragment d'*Infoware* associé à la plate-forme $P3_{(1)}$ devient responsable de la maintenance du VPSN lors d'une mobilité spatiale, au lieu du fragment d'*Infoware* associé à la plate-forme $P2_{(1)}$.

$Event_{(i)}$ 7 : Update VPSN (U_m ; SE2,1$_{(P2)}$; SE2,4$_{(P3)}$; Log@2,4 ; P_3)

$Event_{(i)}$ 8 : Create VPSN & User Profiles (U_m Profile ; $VPSN_m$ Profile)

Comme le montre la Figure 8.14 à t=t_0+2T, le nouveau VPSN de l'utilisateur est constitué, suite au handover sémantique inter-fournisseurs, des deux services : SE1,2$_{(P1)}$ et SE2,4$_{(P3)}$.

D'après cette mise en œuvre organisationnelle, nous avons montré comment les différents services et plates-formes distribués peuvent collaborer afin de gérer le handover sémantique. De plus, nous avons montré comment ce handover sémantique intra-fournisseur ou inter-fournisseurs utilise, tout au long de la mobilité spatiale et temporelle d'un utilisateur, le concept des zones de couverture distribuées afin d'adapter le VPSN de l'utilisateur au contexte ambiant de ce dernier.

9 Conclusion

Dans cette partie, nous avons proposé deux solutions de gestion de la mobilité au niveau "*Service Delivery*". En effet, les solutions existantes que nous avons discutées interviennent toutes au niveau du "*Media Delivery*". Elles gèrent la mobilité de l'utilisateur et du terminal au niveau des réseaux d'accès et au niveau du réseau cœur IP, et elles négligent la couche service. Afin de surmonter cette défaillance, le "*Service Delivery*" agit d'une manière complémentaire au "*Media Delivery*", et il maintient d'une manière dynamique et transparente la continuité de services tout au long de la mobilité spatiale et temporelle de l'utilisateur. Précisément, nous levons deux verrous qui peuvent influencer la continuité de la session de services (VPSN) de l'utilisateur : le premier correspond à la dégradation de la QoS ou le mauvais fonctionnement d'un des services pré-provisionnés dans le VPSN de l'utilisateur ; et le deuxième correspond à la prise en considération du contexte ambiant de l'utilisateur, tout au long de sa mobilité spatiale et temporelle. Pour les lever, nous proposons respectivement une première solution de gestion de mobilité qui s'appuie sur des communautés virtuelles à base d'ubiquité (VSCUs), et puis une deuxième solution de gestion de mobilité qui consiste à étendre le mécanisme de handover du niveau "*Media Delivery*" au niveau "*Service Delivery*" à l'aide d'un mécanisme de handover sémantique. Il faut noter que ces deux solutions sont complémentaires et elles suivent une approche évènementielle à trois rôle : l'initiateur, le décideur et l'exécuteur.

Au niveau de la solution des communautés virtuelles, nous avons montré qu'en combinant des services ubiquitaires ayant la même fonctionnalité et une QoS équivalente dans des communautés virtuelles, et en s'appuyant sur des services autogérés par un agent de QoS intégré dans chacun, nous pouvons anticiper la coupure de la session de service. En effet, lors de la dégradation d'un service, son agent de QoS envoie un *OUT-Contract* pour déclencher le processus de gestion qui consiste à remplacer le service dégrader par un autre qui lui est ubiquitaire et qui appartient à sa propre communauté. Par conséquent, nous maintenons la continuité de services tout en tenant compte des préférences fonctionnelles de l'utilisateur ainsi que de ses préférences non fonctionnelles, c.à.d. sa QoS demandée. Cependant, en plus de l'ubiquité, d'autres intérêts sont pris en considération afin de concevoir d'autres types de communautés virtuelles. En effet,

nous avons conçu des VSC-U&Ps qui combinent des services ubiquitaires appartenant à un même fournisseur, des VSC-U&Ls qui combinent des services ubiquitaires qui sont dans la même zone d'entourage, et enfin, nous avons combiné ces deux types de communautés pour avoir les VSCU-L&Ps. En outre, un autre aspect architectural de notre solution de communautés virtuelles consiste à différencier les Inner VSC-U&Ls des Outer VSC-U&Ls. En effet, une VSC-U&L est considérée comme *Inner* pour le service et la plate-forme qui ont été à l'origine de sa création, et elle est considérée comme *Outer* pour tous les autres services qu'elle a sélectionnés. Par conséquent, une communauté VSC-U&L multi-fournisseurs n'est gérée que par la plate-forme qui l'a créée. Ainsi, nous favorisons dans notre approche une gestion distribuée où chaque plate-forme est responsable de ses propres communautés. Mais, comment chaque plate-forme va-t-elle créer et gérer ses communautés au niveau fonctionnel ? Pour répondre à cette question, nous avons conçu dans chaque plate-forme les services de gestion suivants : le *VSCU Attachment* qui attache un service déployé aux communautés existantes dont il vérifie les caractéristiques (fonctionnalité, QoS, fournisseur, localisation) ; le *VSCU Creation* qui crée l'Inner VSC-U&L relative au service déployé ; l'*External VSCU Maintenance* qui va maintenir les Inner VSC-U&Ls créées par sa plate-forme suite à une dégradation d'un service externe appartenant à ces communautés ; et enfin, l'*Internal VSCU Maintenance* qui va maintenir les Inner VSC-U&Ls créées par sa plate-forme et les VSC-U&Ps relatives à son fournisseur suite à une dégradation d'un service interne.

Au niveau de la solution de handover sémantique, nous avons montré que pour améliorer la QoE de l'utilisateur et diminuer le temps de latence, nous devons adapter d'une manière continue, transparente et dynamique le VPSN de l'utilisateur au contexte ambiant de ce dernier. En effet, notre approche consiste à remplacer un service pré-provisionné dans le VPSN par un autre service ubiquitaire appartenant à la nouvelle zone ambiante de l'utilisateur. Au niveau architectural, nous considérons que chaque fournisseur distribue ses services ubiquitaires dans des zones géographiques à grande échelle de façon à répondre le mieux aux besoins de ses utilisateurs. Ensuite, il associe à chaque service une zone de couverture. Ainsi, lors d'une mobilité spatiale (mobilité du terminal ou mobilité de l'utilisateur), l'utilisateur peut sortir de la zone de couverture relative à l'un de ses services pré-provisionnés et passer à la zone de couverture d'un autre service. Si ce dernier appartient au même fournisseur que le premier, nous considérons qu'il y a un handover sémantique intra-fournisseur, sinon, nous serons dans le cas d'un handover sémantique inter-fournisseurs. Il faut noter que chaque plate-forme est responsable de gérer le handover sémantique de ses propres services. Cependant, la question qui se pose est : comment chaque plate-forme va-t-elle gérer le handover sémantique au niveau fonctionnel ? Pour répondre à cette question, nous avons conçu un ensemble de services : le *SHD*, l'*Intra-SP SHS*, l'*Inter-SP SHS* et le *VPSN Maintenance*. Le *SHD* a pour rôle de détecter si l'utilisateur est sorti de la zone de couverture du service pré-provisionné ou non. Si oui, il initie le handover sémantique en envoyant une notification à l'*Intra-SP SHS*. Ce dernier a pour rôle de décider s'il y a un handover sémantique intra-fournisseur ou non : si oui, il envoie la décision à l'exécuteur (le *VPSN Maintenance*) ; sinon, il envoie la décision à l'*Inter-SP SHS*. Ce dernier a pour rôle de décider s'il y a un handover sémantique inter-fournisseurs ou non : sinon,

il n'y aura pas de handover sémantique pour le service pré-provisionné dans le VPSN ; si oui, l'*Inter-SP SHS* envoie une notification au *VPSN Maintenance* qui exécute la décision prise. Par conséquent, d'après ce processus, l'utilisateur aura toujours dans son VPSN les services qui vérifient non seulement ses préférences fonctionnelles et non fonctionnelles mais aussi son contexte ambiant.

Dans la partie suivante, nous valorisons nos propositions dans le domaine du *cloud computing*. Ensuite, nous démontrons la faisabilité de nos travaux à l'aide d'un démonstrateur et à l'aide d'un scénario applicatif qui met en œuvre l'ensemble des propositions de gestion évoquées.

Troisième partie

Valorisation et Faisabilité

10 Valorisation dans le Cloud

Après avoir présenté nos propositions pour la personnalisation et la convergence de services, pour la gestion de la mobilité et pour la maintenance de la QoS, nous considérons le domaine de *cloud computing* comme le champs d'application le plus impacté par nos travaux et le plus adéquat pour valoriser nos propositions.

En effet, de nos jours, le *cloud computing* devient un des sujets les plus intéressants et les plus traités. Plusieurs définitions ont été données pour décrire ce concept. Cependant, la définition qui combine les aspects primordiaux et qui différencie clairement ce concept des autres paradigmes distribués, est celle donnée par *Prof. Ian Foster* qui est un leader dans les domaines du *grid* et du *cloud computing*. Selon sa définition [26],

"*Cloud computing is a large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the internet*".

D'après cette définition, le *cloud computing* est un paradigm distribué qui s'appuie sur *Internet* afin de rendre ses services d'externalisation. Ces derniers peuvent être des services de calcul, de stockage ou bien même des services réseaux. Cependant, plusieurs questions se posent : l'apogée du *cloud computing* peut il être conservé en se limitant seulement sur la valeur ajoutée de l'externalisation des services ? Comment le *cloud computing* fait-il face aux défis qui dérivent de l'émergence des contextes *user-centric*, NGN et NGS ?

En effet, d'après notre étude, nous croyons qu'au delà de l'externalisation, le *cloud computing* doit prendre en considération les nouveaux besoins émergeant au niveau réseau et au niveau des préférences de l'utilisateur, comme la personnalisation de la session de services, la maintenance de sa QoS ainsi que la gestion de son contexte mobile et ambiant. Cette nouvelle vision du *cloud* permet aux *Cloud Service Providers* (CSPs) d'améliorer leurs ROI en gagnant un plus grand part du marché et en attirant un plus grand nombre d'utilisateurs qui deviennent de plus en plus satisfaits. De plus, du point de vue des utilisateurs du *cloud*, cette approche permet d'enrichir leurs expé-

riences dans ce domaine. Dans cet objectif, notre architecture de services, le NGN/NGS Middleware, peut impacter positivement le monde du *cloud* en apportant des réponses à ses problématiques majeures, comme la personnalisation de services, la gestion de QoS et la mobilité.

Dans la suite, nous discutons les différents travaux qui sont faits afin de répondre à chacun des besoins suivants : la personnalisation de services (voir Section 10.1), la gestion de QoS (voir Section 10.2) et la maintenance de la continuité de services (voir Section 10.3). Ensuite, nous proposons comment nos solutions peuvent améliorer la QoE de l'utilisateur dans le *cloud* ainsi que le rendement de leurs CSPs.

10.1 Personnalisation de services dans le cloud

La personnalisation de services est un des sujets les plus importants pour l'amélioration de l'expérience des utilisateurs dans le *cloud*. Cependant, ce sujet n'est pas assez abordé par les chercheurs et les industriels du *cloud*. Dans cette partie, nous discutons certaines solutions qui ont été proposées pour répondre à ce défi.

Dans [34], les auteurs proposent de fournir des services personnalisés aux utilisateurs en utilisant une plate-forme de services qui est intégrée dans le terminal et qui comporte un ensemble de composants. En effet, le composant *User Information Collector* enregistre les activités et les informations de l'utilisateur pendant des intervalles réguliers, dénommés *Update Cycles*. Ensuite, en se basant sur ces informations, le composant *Model Builder* construit le *User Model* qui sera stocké dans le *User Model Store*. Ainsi, quand l'utilisateur veut partager son modèle avec des CSPs, le composant *Model Synchronizer* envoie le modèle à ces CSPs choisis. Par conséquent, ces derniers auront plus de connaissances sur cet utilisateur et seront capables de fournir des services plus personnalisés.

Dans [30], les auteurs proposent d'envoyer à l'utilisateur des recommandations personnalisées dans le domaine des multi-médias, après avoir agrégé les informations pertinentes qui représentent le contexte de l'utilisateur. Pour ce faire, la proposition se base sur deux parties architecturales : le *Content and Context Information Learner* qui correspond au côté client du service de recommandations, et les composants *cloud* qui correspondent au côté cœur du service de recommandations. La première partie est intégrée dans le terminal et elle s'appuie sur un ensemble de composants comme par exemple le *Local Content Manager* qui identifie le contenu local sur le terminal et le *Context Manager* qui identifie et agrège depuis un calendrier, un GPS, ou une autre source média, le contexte qui s'associe avec une certaine activité de l'utilisateur. La deuxième partie est intégrée dans le *cloud* et elle s'appuie sur un ensemble de composants comme par exemple le *Recommendation Engine Manager* qui contrôle et coordonne les activités et les informations reçues, le *Pattern Analyzer* qui analyse les activités et le comportements, le *Data Miner* qui exploite différentes ressources pour combiner et analyser des informations pertinentes sur l'utilisateur, et le *Recommendation Candidate Generation and Selection* qui génère selon un algorithme spécifique les candidats à recommander.

D'après ces deux articles, nous constatons que la recherche dans le domaine de la personnalisation de services est limitée à la prise en considération du contexte de l'utilisateur lors de sa demande d'un service *cloud*. Autrement dit, l'approche adoptée aujourd'hui représente la personnalisation de services comme une capacité de la part des CSPs à fournir, voir même imposer des services adéquats aux utilisateurs tout en se basant sur des enregistrements analytiques. Cette façon d'aborder le sujet est essentiel, surtout si nous prenons en considération l'étude qui a été faite par le cabinet d'analyse *IDC* [38] et qui montre que "*not enough ability to customize*" est un des cinq défis majeurs dans le monde du *cloud*. Cependant, cette approche n'est pas suffisante, puisqu'elle ne tient pas compte de la personnalisation au niveau de la composition de services.

En effet, dans cette thèse, nous proposons une approche complémentaire à celle évoquée dans ces deux articles précédents. Nous ne nous limitons pas aux recommandations que les CSPs proposent à leurs utilisateurs après avoir analysé leurs profils, leurs activités et leurs contextes. Nous proposons aussi une personnalisation de services qui s'appuie sur une composition dynamique, dans laquelle l'utilisateur impose ses propres préférences fonctionnelles et non fonctionnelles. Ainsi, pour chaque utilisateur, nous associons un VPSN qui pré-provisionne les services *cloud* répondant le mieux à ses préférences. De plus, comme nous l'avons déjà montré, le VPSN s'adapte dynamiquement et d'une manière évènementielle à tout changement dans le contexte ambiant et mobile de l'utilisateur. Afin de supporter cette personnalisation de services, nous nous basons sur des éléments de services *cloud* composables, autonomes, interopérables, *Stateless* et à couplages lâches. Cela permet ainsi de remplacer, dans le VPSN, un service *cloud* pré-provisionné par un autre plus adéquat aux préférences et au contexte de l'utilisateur, sans coupure de la session ouverte.

10.2 Gestion de la QoS dans le cloud

Avec l'émergence de plusieurs CSPs fournissant des services *cloud* identiques, l'utilisateur préfère celui qui garantit la meilleur QoS et qui respecte ses contrats de SLA. Pour cela, la gestion de la QoS dans le *cloud* gagne de plus en plus d'importance. Ci-dessous, nous discutons certains articles qui montrent bien l'intérêt croissant des chercheurs pour cette gestion de QoS dans le monde du *cloud*.

Dans [14], les auteurs proposent un fournisseur *Cloud@Home* qui agrège des ressources hétérogènes offertes par différents fournisseurs *cloud* et les propose d'une manière uniforme à l'utilisateur. Ensuite, il maintient le SLA que signe l'utilisateur avec les fournisseurs de ces ressources. Pour ce faire, le fournisseur *Cloud@Home* fournit une architecture de services qui contient un ensemble de composants pour négocier, contrôler et renforcer la QoS. Cette architecture SOA s'appuie sur trois modules : le *Resource Abstraction Module* qui relie le fournisseur *Cloud@Home* aux différents fournisseurs *cloud* ; le *Resource Management Module* qui collecte les ressources *cloud*, regroupe les requêtes de l'utilisateur et gère les composants de l'architecture *Cloud@Home* ; et enfin, le *SLA Management Module* qui orchestre le processus de gestion de QoS. En effet,

le *SLA Management Module* lance les négociations de QoS entre les utilisateurs du *Cloud@Home* et les fournisseurs *cloud*. Une fois le document de SLA est signé, ce module contrôle les contraintes de QoS. Ainsi, quand il détecte un non respect du contrat, il lance un provisionnement dynamique des ressources qui supportent les préférences de l'utilisateur.

Le principal inconvénient de cette solution est qu'elle est centralisée. En effet, elle s'appuie sur un fournisseur *Cloud@Home* central qui se charge des négociations et de la maintenance de la QoS. Donc, tous les fournisseurs clouds et tous les utilisateurs doivent être reliés à ce même fournisseur *Cloud@Home* qui constitue par conséquence un point de défaillance et d'étranglement qui dégrade les performances.

Dans [44], les auteurs proposent à déléguer le contrôle du SLA à un fournisseur tiers, noté SLMA (*Service Level Management Authority*). Ce dernier est considéré comme un compromis entre les utilisateurs et les fournisseurs des services *cloud* au niveau de la responsabilité de gérer le SLA. Pour ce faire, le SLMA propose trois types de contrôle : le *Synthetic Monitoring*, le *Customer Monitoring* et le *Transaction-based Monitoring*. Le *Synthetic Monitoring* consiste à faire le contrôle uniquement au niveau des fournisseurs. Ainsi, le SLMA détecte les non respects des contrats de SLA seulement quand le service cloud est indisponible ou en panne. Le *Customer Monitoring* consiste à faire le contrôle uniquement au niveau du terminal de l'utilisateur. Ainsi, le SLMA détecte le non respect des contrats de SLA en se basant sur des mesures plus pertinentes et plus complètes de l'environnement *cloud* selon le point de vue des consommateurs. Enfin, le *Transaction-based Monitoring* consiste à utiliser le *HP Business Availability Center* [36]. Ce dernier permet à un fournisseur de contrôler les services métiers tout en prenant en considération le point de vue des consommateurs de ces services. En effet, il contient un ensemble de composants qui gèrent le SLA, contrôlent les transactions IT, identifient les problèmes, les résolvent et les empêchent de se répéter.

Le principal inconvénient de cette solution est qu'elle est centralisée. La plateforme SLMA est centrale, et elle est ainsi considérée comme un point de défaillance et d'étranglement qui dégrade les performances. En outre, la solution de SLMA intervient seulement comme un arbitre entre les fournisseurs *cloud* et les utilisateurs, afin de résoudre le problème de manque de confiance entre ces deux parties. En effet, le SLMA évalue les performances des services *cloud* et détecte les défaillances, mais il ne propose aucune solution pour surmonter ces problèmes de SLA.

Afin de dépasser ces inconvénients et ces insuffisances au niveau de ces solutions discutées, le *cloud* a besoin d'une solution décentralisée qui ne se limite pas à évaluer les performances et à détecter les défaillances des services *cloud*, mais à répondre aussi d'une manière dynamique et transparente à ces dégradations de QoS. Pour cette raison, nous assurons que les solutions que nous avons proposées dans cette thèse au niveau de la gestion de la QoS peuvent influencer positivement les performances du *cloud* et assurent un grand pas vers un *cloud* plus efficace. En effet, grâce à notre modèle de QoS à quatre critères (disponibilité, capacité, délai et fiabilité), nous présentons d'une manière homogène la partie non fonctionnelle des services *cloud*, ce qui facilite la gestion de QoS et la rend plus rapide et plus flexible. En outre, d'après nos solutions, nous favorisons une gestion décentralisée au cours de laquelle, il n'y a pas un point de

défaillance et d'étranglement qui peut causer une dégradation des performances. Notre approche décentralisée se base sur des services autogérés, dont chacun est responsable de sa propre QoS et participe à l'anticipation de la coupure de la session de services. Pour ce faire, un agent de QoS est associé à chaque service, ce qui lui permet de contrôler sa propre QoS et de notifier par un *OUT-Contract* les autres services concernés, lors d'une dégradation ou d'un mauvais fonctionnement. Afin de permettre aux fournisseurs *cloud* de réagir dynamiquement à ces dégradations et de mieux respecter les préférences de leurs utilisateurs, nous avons proposé un nouveau concept qui consiste à combiner des services ubiquitaires, ayant une même fonctionnalité et une QoS équivalente, dans des communautés virtuelles. Ainsi, ces dernières proposent des services qui sont capables de remplacer le service dégradé d'une manière continue et transparente, ce qui permet de respecter la QoE de l'utilisateur et de garantir ses préférences fonctionnelles et non fonctionnelles.

10.3 Gestion de la Mobilité dans le cloud

D'après une étude qu'elle a faite en 2009, *ABI Research* [1] prévoit qu'en 2015, il y aura plus que 240 millions consommateurs métier qui vont se servir de leurs propres équipements afin d'utiliser les services du *cloud computing*, ce qui va assurer des revenus de l'ordre de 5.2 billions de dollars. Il faut noter que, selon *ABI Research* [1], le nombre des abonnés dans le domaine du MCC (*Mobile Cloud Computing*) était seulement 42.8 millions en 2008. Avec cette émergence du MCC, la latence commence à se poser comme une problématique majeure pour le *cloud*. Ce dernier se base sur l'externalisation des services ainsi que de leurs données respectives dans des serveurs *cloud* qui sont normalement placés loin des utilisateurs. Ainsi, une forte latence s'impose normalement quand l'utilisateur désire accéder à ces services *cloud*, ce qui peut causer une dégradation de sa QoE. En outre, ce problème de latence s'aggrave encore plus avec l'évolution des *smartphones* qui constituent un support primordial pour le marché du MCC.

Afin de dépasser cette problématique, la notion du *Cloud Service Placement* [54] doit être considérée comme un défi fondamental dans le contexte MCC. Il faut la discuter et essayer de la résoudre pour pouvoir améliorer la QoE des utilisateurs dans le *cloud*. Dans ce but, plusieurs technologies ont été conçues. Par exemple, nous citons l'émergence des *Location-based Services* [67][64] qui améliorent la prise de conscience du contexte de l'utilisateur. Il y a aussi l'*IBM Blue Cloud* [37] qui consiste à concevoir des *Data Centers* distribués pour un meilleur passage à l'échelle et une tolérance plus efficace aux pannes. L'*IBM Blue Cloud* garantit un traitement distribué des données à l'aide de ressources distribuées globalement accessibles. En outre, *Amazon* a aussi conçu sa propre solution distribuée au niveau du "*Content Delivery*" dans le *cloud*. Elle s'appelle *Amazon CloudFront* [4]. Cette solution consiste à fournir les contenus statiques et les contenus de *streaming* à l'aide d'un réseau global d'*Edge Locations*. Ainsi, les requêtes sont dynamiquement routées vers l'*Edge Location* le plus proche, ce qui permet de fournir le contenu avec la moindre latence et la meilleure performance possible.

D'après ces exemples cités, nous pouvons assumer que les CSPs sont conscients de l'influence de la latence et du contexte ambiant sur la QoE de l'utilisateur. Pour cette raison, ils ont conçus des *Location-based Services* et des *Data Centers* distribués. Cependant, cette nouvelle tendance au niveau MCC se limite à un "*Content and Data Delivery*" distribué pour certaines applications mobiles.

Afin de dépasser cette limitation, nous proposons de commencer par différencier le "*Service Delivery*" du "*Content and Data Delivery*". Ensuite, nous proposons d'étendre le concept de distribution vers cette couche de "*Service Delivery*". Par conséquent, nous proposons que les CSPs déploient des services ubiquitaires dans différentes zones géographiques afin de répondre au mieux aux demandes des utilisateurs, en diminuant la latence et en personnalisant les sessions de services de ces utilisateurs par rapport à leurs contextes ambiants. Cette adaptation de la session de services d'un utilisateur se fait tout au long de la mobilité spatiale et temporelle de ce dernier, et elle s'effectue par chacun des CSPs auxquels l'utilisateur est abonné. Elle s'appuie sur le handover sémantique afin de remplacer dynamiquement et sans couture un service *cloud* pré-provisionné par un autre qui vérifie le nouveau contexte ambiant de l'utilisateur. Ainsi, cela permet d'améliorer la QoE et la prise de conscience du contexte ambiant de l'utilisateur. Un autre avantage de notre approche est que les requêtes provenant de différents utilisateurs pour un même service seront exécutées par différents services ubiquitaires distribués qui vérifient les différentes zones de localisation de ces utilisateurs. Par conséquent, cette approche permet de dépasser les problèmes des solutions centralisées, comme la surcharge du trafic, la surcharge du traitement et l'étranglement.

11 Faisabilité

Dans ce chapitre, nous présentons d'abord dans la Section 11.1 un scénario applicatif qui montre la gestion de la continuité de la session de services d'un utilisateur dans le cas d'une mobilité spatiale et temporelle. Pour mettre en œuvre ce scénario, nous détaillons dans la Section 11.2 notre démonstrateur UBIS qui correspond à une plate-forme de développement des services applicatifs et des services de gestion.

11.1 Scénario Applicatif

Nous présentons dans cette section un cas d'usage de la gestion de la session de services d'un utilisateur donné dans un contexte NGN/NGS mobile et ubiquitaire.

Nous prenons le cas d'Alice qui est une cliente UBIS qui désire échanger sa maison, pour ses vacances, avec une autre personne habitant à Nice. Pour ce faire, elle consulte son catalogue de service exposables, ainsi que les services exposables qui sont découverts par son terminal. Parmi les différents offres possibles, elle décide de s'abonner, selon ses préférences, aux deux services exposables suivants : le service d'échange de maison pour les vacances, *HomeSwap_STG*, qui est fourni par la société *SwapToGo* et le service d'appel-visio, *FaceTime_SFR*, qui est fourni par *SFR*.

En effet, le service *HomeSwap_STG* permet de sélectionner quelques annonces de gens qui échangent leurs maisons pour les vacances à Nice, de récupérer leurs contacts, de visualiser et d'échanger les photos des maisons avec les contacts choisis, et enfin, de visualiser la localisation géographique de ces maisons. Pour cela, nous considérons que le service *HomeSwap_STG* est décomposable en quatre éléments de services : *HomeAnnouncement_STG*, *Contact_STG*, *Gallery_Flickr* et *Map_Mappy*. Nous constatons que ces éléments de services ne sont pas tous fournis par la société *SwapToGo* ; quelques uns sont fournis par d'autres fournisseurs partenaires. En outre, le service de *FaceTime_SFR* est décomposé en deux éléments de services : *Voice_SFR* et *Video_SFR*. Par conséquent, Alice obtient un VPSN qui sélectionne cinq éléments de services (voir Figure 11.1) :

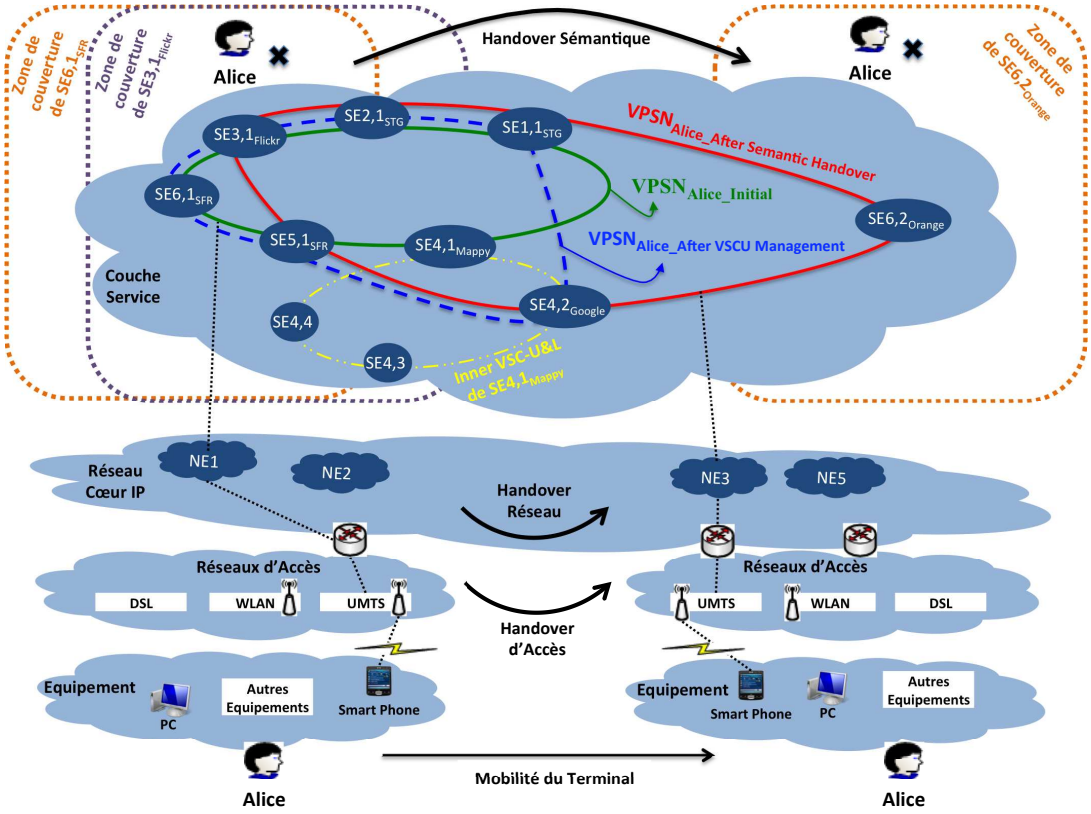

Fig. 11.1: Scénario applicatif.

$$
\begin{aligned}
\mathit{VPSN\_Alice\_Initial} : \; & \mathit{HomeAnnouncement\_STG} \text{ (noté } \mathit{SE1,1_{STG}}) \\
& + \mathit{Contact\_STG} \text{ (noté } \mathit{SE2,1_{STG}}) \\
& + \mathit{Gallery\_Flickr} \text{ (noté } \mathit{SE3,1_{Flickr}}) \\
& + \mathit{Map\_Mappy} \text{ (noté } \mathit{SE4,1_{Mappy}}) \\
& + \mathit{Voice\_SFR} \text{ (noté } \mathit{SE5,1_{SFR}}) \\
& + \mathit{Video\_SFR} \text{ (noté } \mathit{SE6,1_{SFR}}).
\end{aligned}
$$

Au cours de la journée, l'agent de QoS associé au service *Map_ Mappy* a détecté une dégradation de la QoS courante qui passe au dessous de la QoS seuil permise. Ainsi, il envoie un *OUT-Contract* dans l'Inner VSC-U&L correspondante (VSC-U&L$_{(4,1)}$). Cette dernière contient les services ubiquitaires au service *Map_ Mappy* et qui sont capables de le remplacer. Nous supposons que parmi ces services ubiquitaires, le service *GoogleMaps_ Google* (noté $\mathit{SE4,2_{Google}}$) est sélectionné pour remplacer *Map_ Mappy* dans le *VPSN_ Alice*. Ce dernier devient sous la forme suivante (voir Figure 11.1) :

$$
\begin{aligned}
\mathit{VPSN\_Alice\_After\ VSCU\ Management} : \; & \mathit{HomeAnnouncement\_STG} \text{ (noté } \mathit{SE1,1_{STG}}) \\
& + \mathit{Contact\_STG} \text{ (noté } \mathit{SE2,1_{STG}}) \\
& + \mathit{Gallery\_Flickr} \text{ (noté } \mathit{SE3,1_{Flickr}}) \\
& + \mathit{GoogleMaps\_Google} \text{ (noté } \mathit{SE4,2_{Google}}) \\
& + \mathit{Voice\_SFR} \text{ (noté } \mathit{SE5,1_{SFR}}) \\
& + \mathit{Video\_SFR} \text{ (noté } \mathit{SE6,1_{SFR}}).
\end{aligned}
$$

Par conséquent, le fournisseur du service *Map_ Mappy* a pu anticiper la coupure de la session de services, en remplaçant d'une manière dynamique son service dégradé par un autre service ubiquitaire. Cependant, il faut noter que la continuité du service *Map_ Mappy* n'aura pas pu être garantie, si son fournisseur n'avait pas bien maintenue en avance l'Inner VSC-U&L correspondante.

Nous supposons maintenant qu'Alice est sur son chemin vers son travail, et elle a sa session de services toujours ouverte sur son *smartphone*. Ainsi, une mobilité du terminal est détectée par ce dernier lors de chaque vérification périodique de la localisation géographique. Par conséquent, le terminal notifie toutes les plates-formes responsables des services sélectionnés dans le VPSN d'Alice. Chaque *SHD* relatif à chacune de ces plates-formes vérifient si Alice est sortie de la zone de couverture qui correspond au service relatif à sa plate-forme, afin d'initier un handover sémantique. Par exemple, le *SHD* relatif à la plate-forme qui fournit le service *Voice_ SFR* vérifie si Alice est sortie ou pas de la zone de couverture de ce service *Voice_ SFR*.

Pour simplifier le schéma présenté dans la Figure 11.1, nous montrons une seule vérification qui est faite au bout d'une période *n*T*. Nous considérons aussi que seul les *SHDs* qui correspondent aux plates-formes des services *Gallery_ Flickr* et *Video_ SFR* ont détecté une sortie d'Alice des zones de couvertures correspondantes. Ainsi, en parallèle aux handovers qui s'effectuent au niveau des réseaux d'accès et du réseau cœur, les *SHDs* initient chacun un handover sémantique au niveau de la couche service.

Pour ce faire, chaque *SHD* envoie une notification vers l'*Intra-SP SHS* de sa plate-forme. Ensuite, chaque *Intra-SP SHS* vérifie s'il y a un service ubiquitaire à son service géré (*Gallery_ Flickr* ou bien *Video_ SFR*) dans la VSC-U&P qui correspond à ce dernier. Nous prenons le cas où aucun service ubiquitaire des VSC-U&Ps des services *Gallery_ Flickr* et *Video_ SFR* n'a une zone de couverture qui vérifie la nouvelle localisation d'Alice. Par conséquent, chacun des deux *Intra-SP SHS* décide qu'il n'y a pas un handover sémantique intra-fournisseur, et envoie une notification vers l'*Inter-SP SHS* correspondant afin que ce dernier vérifie s'il y a un handover sémantique inter-fournisseurs ou non.

Pour le cas du service *Gallery_ Flickr*, l'*Inter-SP SHS* relatif à sa plate-forme vérifie si Alice se trouve dans une des zones de couverture des services sélectionnés dans l'Inner VSC-U&L du service *Gallery_ Flickr*. Après la comparaison, l'*Inter-SP SHS* ne trouve aucun service ubiquitaire dont la zone de couverture vérifie la nouvelle localisation d'Alice. Ainsi, l'*Inter-SP SHS* décide de ne pas effectuer un handover sémantique pour le service *Gallery_ Flickr* qui reste ainsi dans le VPSN d'Alice.

Pour le cas du service *Video_ SFR*, l'*Inter-SP SHS* relatif à sa plate-forme vérifie si Alice se trouve dans une des zones de couverture des services sélectionnés dans l'Inner VSC-U&L du service *Video_ SFR*. Normalement, d'une part, l'utilisateur désire avoir une bonne qualité vidéo, pour cela il préfère être relié au service vidéo le plus proche, celui qui assure le plus faible temps de latence pour sa réponse. D'autre part, les fournisseurs des services vidéo désirent satisfaire le plus grand nombre d'abonnés, pour cela ils déploient plusieurs services vidéo ubiquitaires dans différentes plates-formes distribuées sur plusieurs zones géographiques. Par conséquent, dans le cas du service

Video_ SFR, nous considérons que son *Inter-SP SHS* va normalement trouver un service ubiquitaire dont la zone de couverture vérifie la nouvelle localisation d'Alice. En effet, parmi les services de l'Inner VSC-U&L de *Video_ SFR*, l'*Inter-SP SHS* trouve qu'Alice se trouve dans la zone de couverture du service *Video_ Orange* qui est fourni par la société *Orange*. Ainsi, le service *Video_ Orange* remplace le service *Video_ SFR* dans le VPSN d'Alice. Ce dernier devient sous la forme suivante (voir Figure 11.1) :

VPSN_ Alice_ After Semantic Handover : *HomeAnnouncement_ STG* (noté $SE1{,}1_{STG}$)
+ *Contact_ STG* (noté $SE2{,}1_{STG}$)
+ *Gallery_ Flickr* (noté $SE3{,}1_{Flickr}$)
+ *GoogleMaps_ Google* (noté $SE4{,}2_{Google}$)
+ *Voice_ SFR* (noté $SE5{,}1_{SFR}$)
+ *Video_ Orange* (noté $SE6{,}2_{Orange}$).

Suite à ce scénario applicatif, nous avons pu montrer comment la session de services d'un utilisateur est maintenue d'une manière continue, dynamique et transparente, tout au long de la mobilité temporelle de ce dernier, suite à une dégradation de la QoS ou un mauvais fonctionnement d'un des services sélectionnés dans le VPSN et lors d'une mobilité spatiale de la part de cet utilisateur. La gestion de mobilité traitée dans ce scénario applicatif, est mise en œuvre dans la suite par un démonstrateur, dans lequel nous intégrons tous les services exposables et les services de gestion nécessaires.

11.2 Démonstrateur

Pour mettre en œuvre nos propositions, nous nous appuyons sur le démonstrateur du projet UBIS dans lequel se situe notre travail. En effet, la Sous-Section 11.2.1 représente l'architecture de ce démonstrateur, et la Sous-Section 11.2.2 représente les automates des composants de services développés.

11.2.1 Architecture du démonstrateur

Comme le montre la Figure 11.2, ce démonstrateur intervient au niveau des différentes couches de visibilité : utilisateur, terminal, transport, contrôle et service. Il traite les aspects de signalisation, de gestion, de sécurité et de maintenance des informations dans les bases de connaissances.

Au niveau utilisateur, l'approche *user-centric* consiste à prendre en considération toutes les préférences de chaque utilisateur, ainsi que ses besoins et ses abonnements. Pour ce faire, un *User Profile* est créé et maintenue par chacune des bases de connaissances : les *Infoware* qui sont associés aux plates-formes auxquelles l'utilisateur s'est abonné, et l'*Infosphère* qui est associée au terminal de ce dernier.

Au niveau du terminal, l'approche *user-centric* nécessite une plate-forme qui définit des fonctionnalités de personnalisation et d'adaptation tout en tenant compte des préférences de l'utilisateur ainsi que son contexte ambiant. Pour ce faire, un *Userware* est intégré dans chaque terminal. Il agira comme un marché ouvert de services où les utilisateurs pourront vendre (offre) et/ou acheter (demande) les services, les uns des

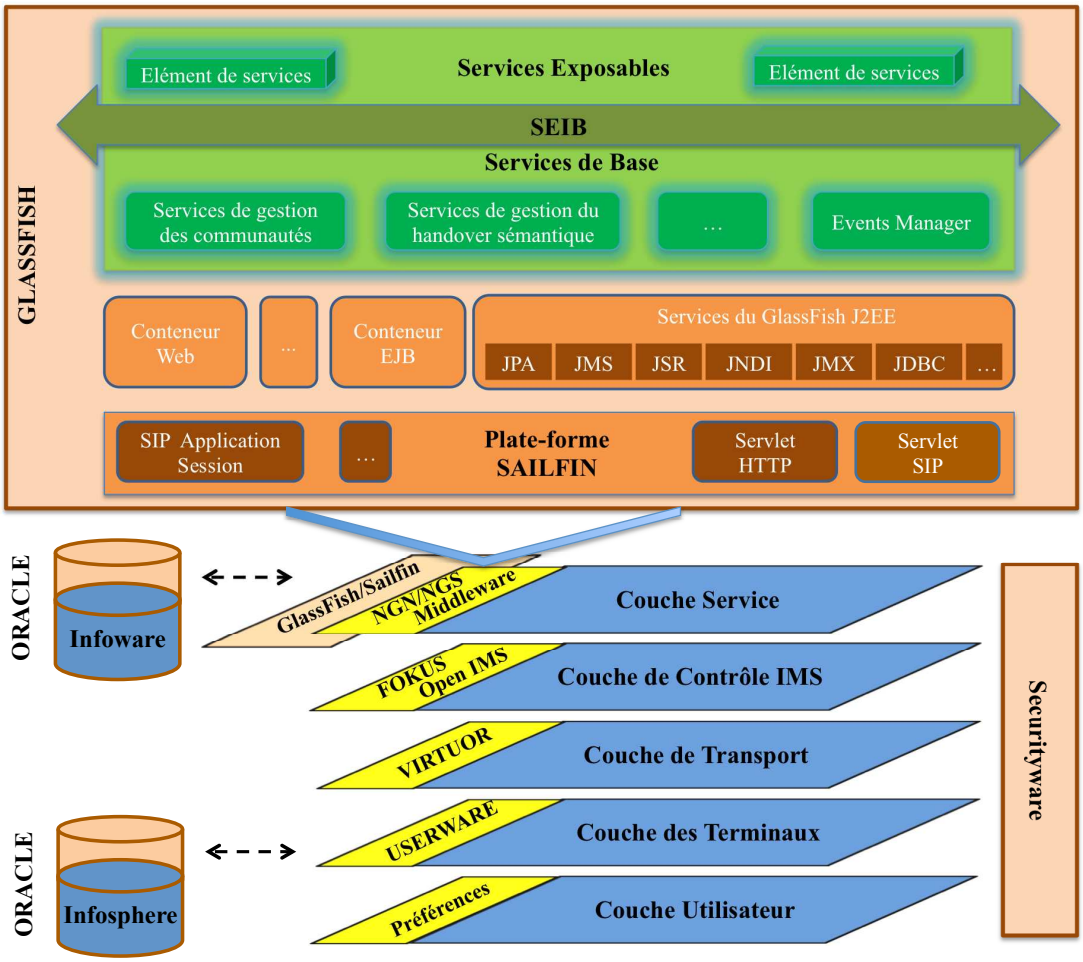

Fig. 11.2: Architecture du Démonstrateur sur différents niveaux de visibilité.

autres. De plus, il fournit l'ensemble des services nécessaires pour permettre un accès personnalisé aux services exposables demandés.

Au niveau de la couche de transport, l'approche *user-centric* nécessite une adaptation dynamique et transparente des réseaux de transport suite à toute modification faite au niveau service. Pour ce faire, nous faisons appel à la virtualisation qui est introduite au niveau réseau par la solution *VirtuOR* [65]. Cette dernière propose l'intégration de plusieurs entités virtuelles sur une même machine physique. Ces entités virtuelles peuvent être des routeurs, des points d'accès virtuels, des serveurs SIP, etc. Ainsi, *VirtuOR* garantit l'accès à chaque service en lui associant un réseau virtuel spécifique.

Au niveau de la couche de contrôle IMS, nous utilisons l'*Open IMS* de *FOKUS* [25]. Cette couche de contrôle est introduite entre les couches accès/transport d'une part et la couche de services de l'autre part. Son rôle principal est de contrôler les sessions ouvertes (établissement, modification et libération des sessions multimédias), en se basant sur le protocole de signalisation SIP. Afin de tenir compte de l'approche *user-centric*, et de maintenir une session continue et personnalisée, nous utilisons un protocole SIP+ qui est une version évoluée du protocole SIP.

Au niveau informationnel, nous ne pouvons pas nous contenter du HSS qui est introduit par le *FOKUS Open IMS*, puisqu'il constitue une simple base de données ayant un simple rôle de stockage d'informations. Or, l'approche *user-centric* nécessite d'avoir des bases de connaissances qui interviennent selon des inférences informationnelles afin de tenir compte dynamiquement et en temps réel de l'information personnalisée et des ressources ambiantes. Dans notre démonstrateur, nous associons un *Infoware* à chaque plate-forme d'un fournisseur de services, et un *Infosphère* à chaque terminal d'un utilisateur donné. Afin de mettre en place nos bases de connaissances, nous avons effectué une migration de la base de données *MySQL* du HSS de *FOKUS* en une base de données *Oracle 11g* [49].

Au niveau de la sécurité, nous maintenons la sécurité de bout-en-bout en utilisant une plate-forme de services spécifique, nommée *Securityware*. Cette dernière propose un ensemble de services de sécurité qui traitent l'authentification, l'autorisation, la fédération et l'audit. De plus, un *Policy Agent* est introduit dans chaque NGN/NGS Middleware et dans chaque terminal afin que chaque plate-forme et chaque terminal puissent contacter le *Securityware*. Il faut noter que les services de sécurité sont développés de la même façon que les autres services de base et les services exposables.

Au niveau de la couche service, nous avons intégré notre NGN/NGS Middleware dans un serveur *GlassFish* [61]. Ce dernier est choisi puisqu'il constitue une plate-forme de services *open source*, standard, facilement utilisée pour construire et déployer des services de nouvelle génération (NGS), et idéale pour les architectures orientées services (SOA). De plus, *GlassFish* est le premier serveur à soutenir la technologie *Java Enterprise Edition 6* qui fournit un accès rapide à une plate-forme sécurisée, portable et évolutive pour les services d'entreprise, et qui supporte différentes APIs comme JMS (*Java Message Service*), JNDI (*Java Naming and Directory Interface*) et JDBC (*Java Database Connectivity*), etc. Au niveau de la signalisation, un serveur *Sailfin* [63] est aussi intégré dans la plate-forme *GlassFish*. L'objectif de *Sailfin* est de permettre au serveur *GlassFish* de répondre aux besoins de communication et des applications multimédias. De plus, il aide les fournisseurs de services et les opérateurs de réseau à accélérer l'adoption d'une architecture IMS, en leur offrant une plate-forme standardisée qui simplifie le développement des applications et des services à valeurs ajoutées. En effet, *Sailfin* combine l'architecture SOA des entreprises et les capacités des services Web avec des *HTTP Servlets* et des *SIP Servlets*. Ces derniers sont à l'origine de nombreux services populaires, comme la voix sur IP, la messagerie instantanée, les conférences web, etc. De plus, ils jouent un rôle encore plus important dans la construction de la prochaine génération de services de télécommunications. Il faut noter que la plate-forme de services communique au niveau d'usage à l'aide du HTTP et au niveau de la signalisation à l'aide du SIP amélioré (SIP+).

Au sein du serveur *GlassFish*, nous avons validé nos propositions en développant chaque service de gestion et chaque service exposable sous forme d'un EJB (*Enterprise Java Bean*) indépendant [60]. Nous utilisons les EJBs car ils permettent la création de différents composants autonomes et connectés à travers des liens lâches. En outre, les EJBs fournissent une transparence pour les transactions distribuées et aident à la création de solutions extensibles et à longue portée.

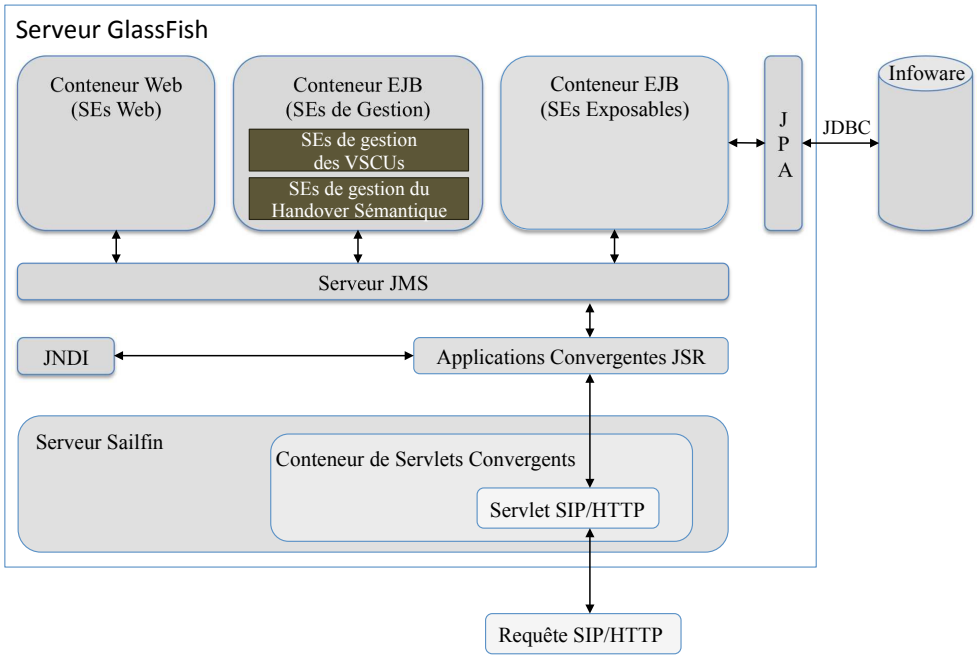

Fig. 11.3: Aspects fonctionnels du démonstrateur au niveau service.

D'après la Figure 11.3, nous montrons les composants fonctionnels qui permettent d'intégrer notre NGN/NGS Middleware dans le serveur *GlassFish/Sailfin*. Il y a un conteneur Web pour les services Web, un conteneur EJB pour les services exposables et un un conteneur EJB pour nos services de gestion, y compris ceux pour la gestion du handover sémantique et ceux pour la gestion par les communautés virtuelles. Afin de supporter une interaction asynchrone entre les composants de services développés, nous utilisons le serveur JMS intégré dans la plate-forme *GlassFish*. En effet, JMS est une API qui permet d'envoyer et de recevoir des messages de manière asynchrone entre des composants Java. Afin de relier les services développés aux données relationnelles stockés dans la base informationnelle, nous utilisons la technologie JPA (*Java Persistence API*). Cette dernière s'appuie sur un ensemble de fichiers qui correspondent à des classes d'entités de persistance, et qui contiennent des références vers les connecteurs JDBC à utiliser. Ainsi, le JDBC fournit les méthodes pour lancer les requêtes et les mises à jour pour les données dans la base. Afin de gérer les noms des files d'attente associées à chaque composant de service, nous utilisons le *JNDI Directory*. Ce dernier est un annuaire qui introduit des fonctionnalités de nommage permettant d'associer un nom à chaque service. Ainsi, chaque service développé utilise le JNDI pour se connecter à l'annuaire et trouver le service suivant dans la logique de services. Enfin, pour que les services puissent répondre aux requêtes envoyées par l'utilisateur (des requêtes SIP pour la signalisation et le *provisioning* et des requêtes HTTP pour l'usage), nous utilisons le *JSR Converged Applications*. Ce dernier relie les conteneurs de services et les conteneurs des *Servlets*. En effet, les SIP *Servlets* sont intégrés dans *Sailfin* pour former des interfaces pour les requêtes SIP, et les HTTP *Servlets* sont intégrés dans *Sailfin* pour former des interfaces pour les requêtes HTTP.

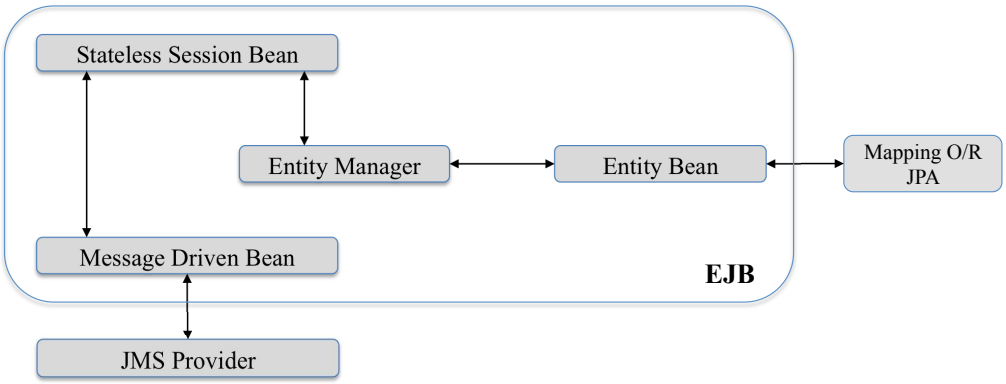

Fig. 11.4: Structure d'un EJB.

Après avoir décrit le rôle de chacun des différents composants fonctionnels utilisés pour l'intégration de nos composants de services, nous présentons dans la Figure 11.4 la structure fonctionnelle d'un EJB. En effet, chaque composant de service implémenté en EJB respecte la structure suivante :

- *Message Driven Bean* : il permet le traitement des messages asynchrones. Il reste à l'écoute de la file d'attente de JMS et réagit dès l'arrivée d'une requête. Ainsi, il extrait cette dernière de la file d'attente, en récupère le contenu puis exécute un traitement.
- *Stateless Session Bean* : il est un *bean* qui tient des "conversations" à un seul appel de méthodes avec des utilisateurs et avec d'autres EJBs. Il n'exige pas que l'état soit maintenu à travers les différentes requêtes. Par conséquent, le *Stateless Session Bean* permet la réutilisation et la mutualisation du composant. Il faut noter que ces *Stateless Session Beans* sont utilisés afin d'implémenter nos composants *Stateless* mais ils interfèrent en sessions *Statefull*.
- *Entity Bean* : il est un objet avec un domaine de persistance léger. Il représente un ou plusieurs tables dans notre base de données relationnelles.
- *Entity Manager* : il est utilisé afin de créer et d'enlever des entités persistantes, de trouver ces entités par leurs clés primaires, et d'envoyer des requêtes à la base de données afin de s'informer sur ces entités.

11.2.2 Automates des composants de services

Dans cette sous-section, nous représentons sous forme d'automate chacun de nos services de gestion de mobilité : le *VSCU Attachment* (voir Figure 11.5), le *VSCU Creation* (voir Figure 11.6), l'*Internal VSCU Maintenance* (voir Figure 11.7), l'*External VSCU Maintenance* (voir Figure 11.8), le *SHD* (voir Figure 11.9), l'*Intra-SP SHS* (voir Figure 11.10), l'*Inter-SP SHS* (voir Figure 11.11) et le *VPSN Maintenance* (voir Figure 11.12).

Chacun de nos services de gestion et de nos services exposables agit sur trois plans : le plan d'usage, le plan de gestion et le plan de contrôle. Le plan d'usage regroupe les fonctions principales réalisées par le composant pour rendre son service. Le plan

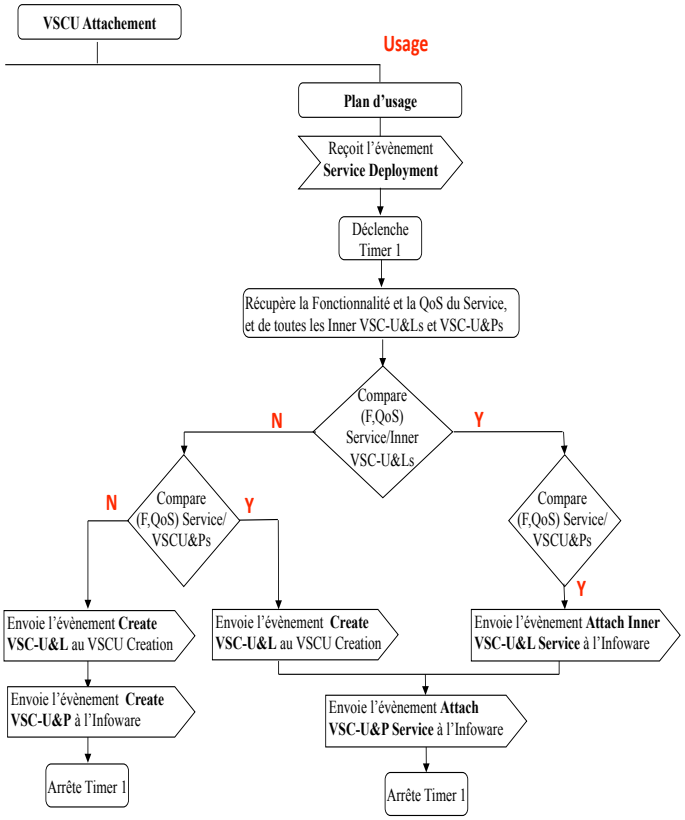

Fig. 11.5: Automate du *VSCU Attachment*.

de contrôle inclut les procédures de signalisation relatives à l'activation/désactivation du composant, provisionnement du composant et exécution ou interruption de son fonctionnement. Le plan de gestion regroupe les fonctions d'autogestion effectuées par l'agent de QoS qui est associé au composant de service. Parmi ces fonctions de gestion, il y a la vérification de la QoS courante, les échanges des *IN-Contracts* et *OUT-Contracts* qui se produisent entre les composants de services appartenant à une même communauté pour la gestion du VPSN au bout d'un OUT-Contract et pour la gestion de cette communauté au bout de trois *OUT-Contracts* consécutifs.

Dans les automates de nos services de gestion, nous nous focalisons seulement sur le plan d'usage ; les autres plans ne rentrent pas dans le cadre de cette thèse. Comme nous le montrons dans ces différents automates, le plan d'usage commence par un déclenchement d'un *Timer 1*. Ce dernier représente le temps nécessaire pour chaque service pour finir le traitement de chaque requête reçue. Si ce *Timer 1* expire, l'agent de QoS de ce service intervient au niveau du plan de gestion afin d'envoyer un *OUT-Contract* qui déclenche le processus de maintenance du VPSN. De plus, un *Timer 2* apparaît aussi dans les plans d'usage des services *VSCU Creation* et *External VSCU Maintenance*. Ce *Timer 2* est déclenché lorsque ces services font appel au service de découverte *DBS*. S'il expire, les agents de QoS associés aux services *VSCU Creation* et *VSCU Maintenance* doivent intervenir afin de trouver un autre service DBS qui peut répondre aux requêtes, sûrement avant l'expiration du *Timer 1*.

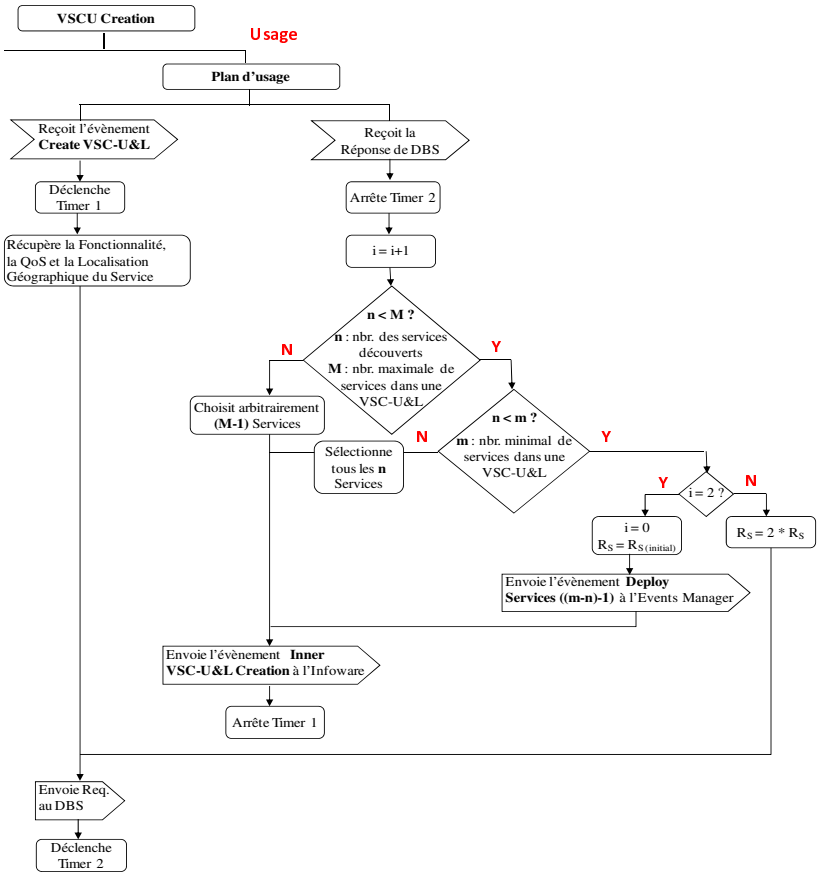

Fig. 11.6: Automate du *VSCU Creation*.

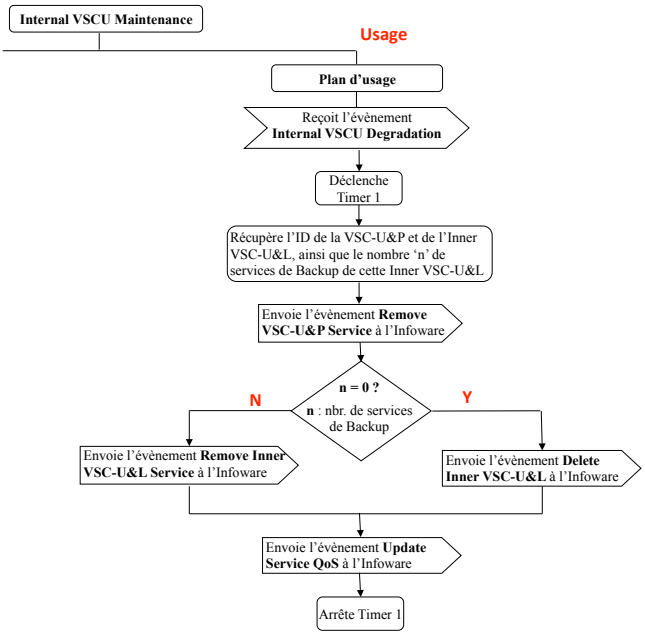

Fig. 11.7: Automate de l'*Internal VSCU Maintenance*.

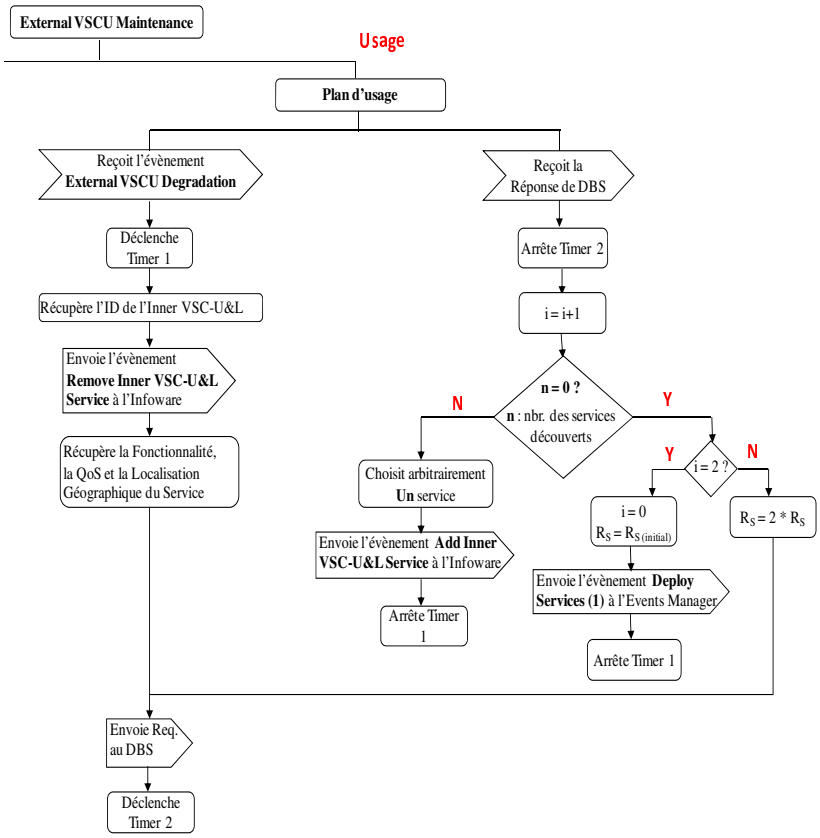

Fig. 11.8: Automate de l'*External VSCU Maintenance*.

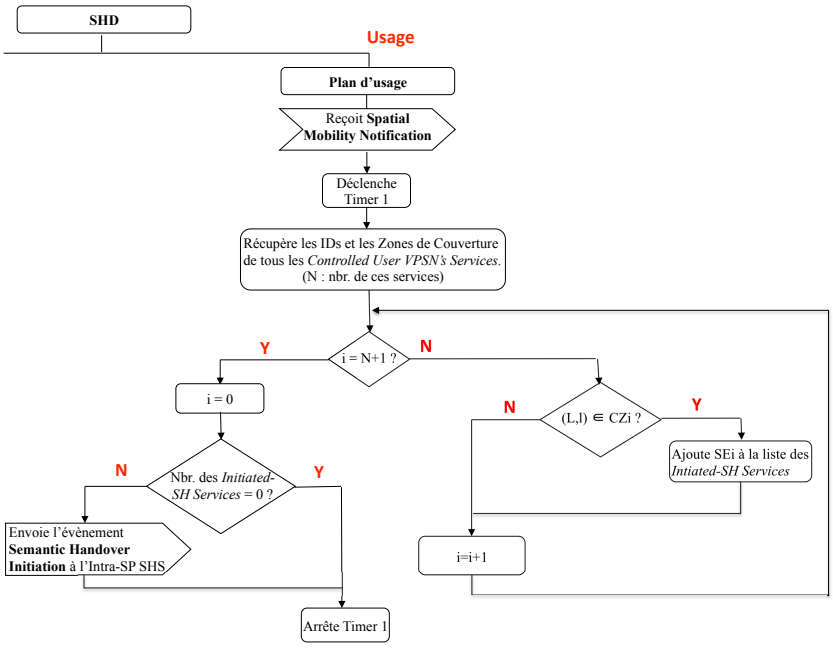

Fig. 11.9: Automate du *SHD*.

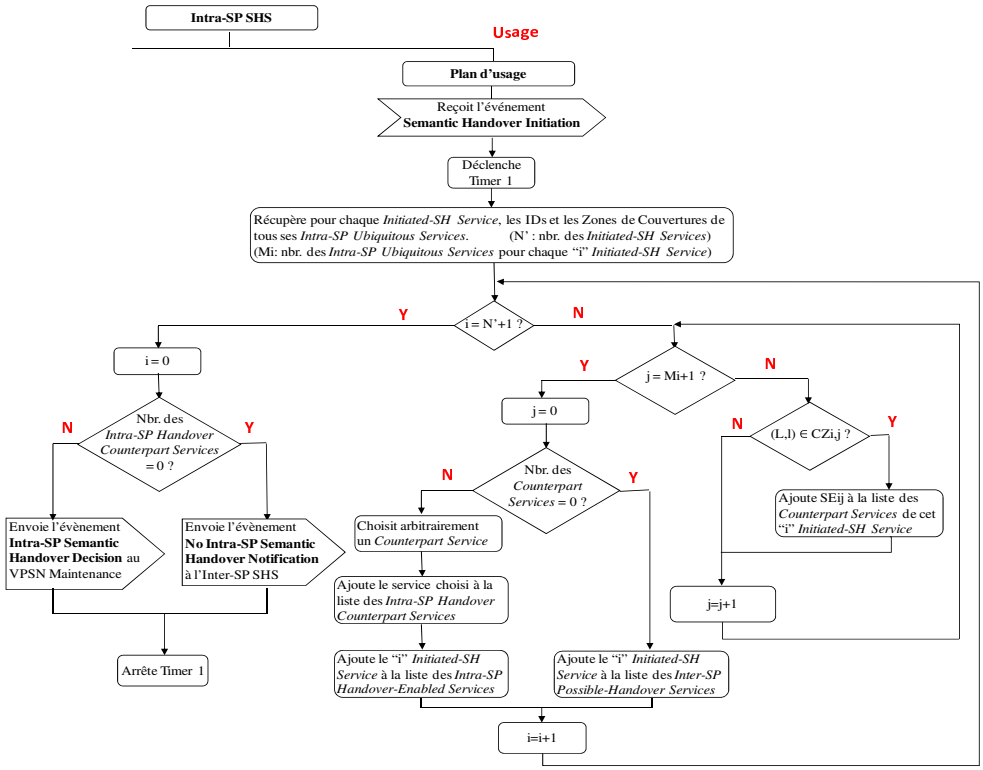

Fig. 11.10: Automate de l'*Intra-SP SHS*.

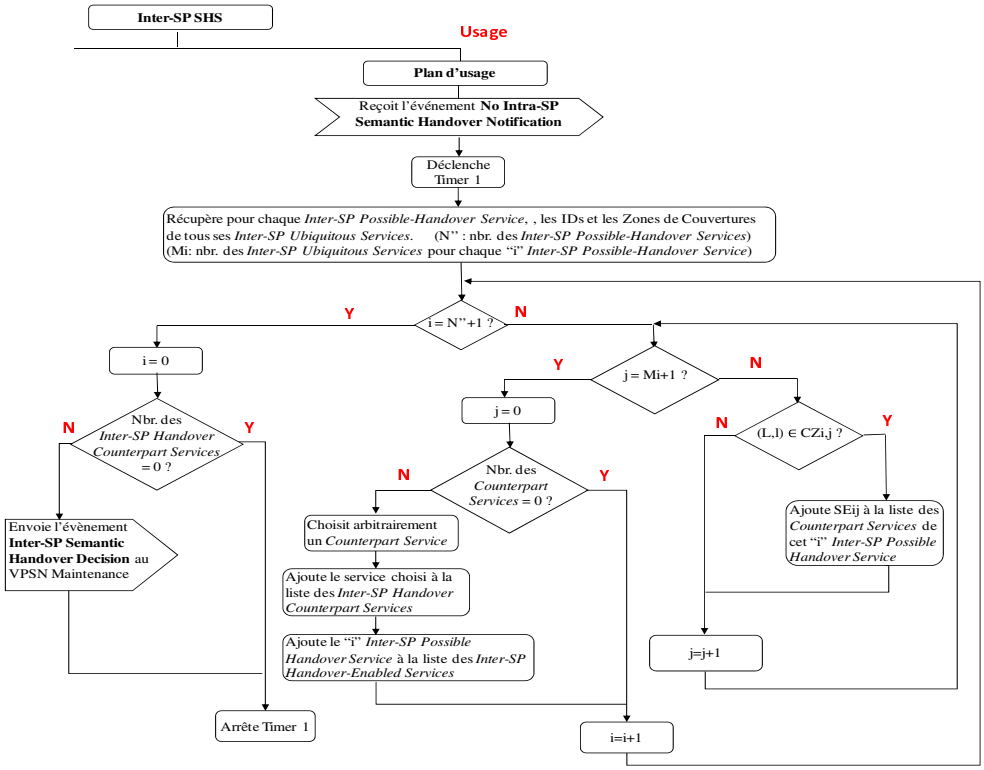

Fig. 11.11: Automate de l'*Inter-SP SHS*.

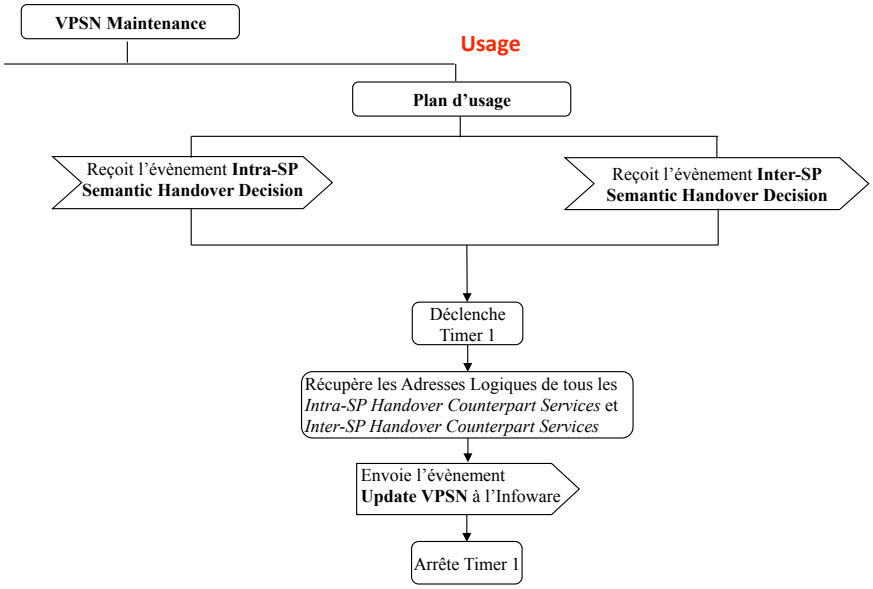

Fig. 11.12: Automate du *VPSN Maintenance*.

Pour conclure, ces services de gestion de mobilité, qui sont représentés dans ces automates, sont déployés sous forme d'EJBs de base dans le démonstrateur. De plus, les services exposables qui sont utilisés dans le scénario applicatif seront eux aussi déployés dans le démonstrateur sous forme d'EJBs applicatifs. La Figure 11.13 représente les deux conteneurs qui contiennent les services exposables et les services de gestion de mobilité. Ce démonstrateur permettra ainsi de mettre en œuvre le scénario applicatif et de valider nos propositions.

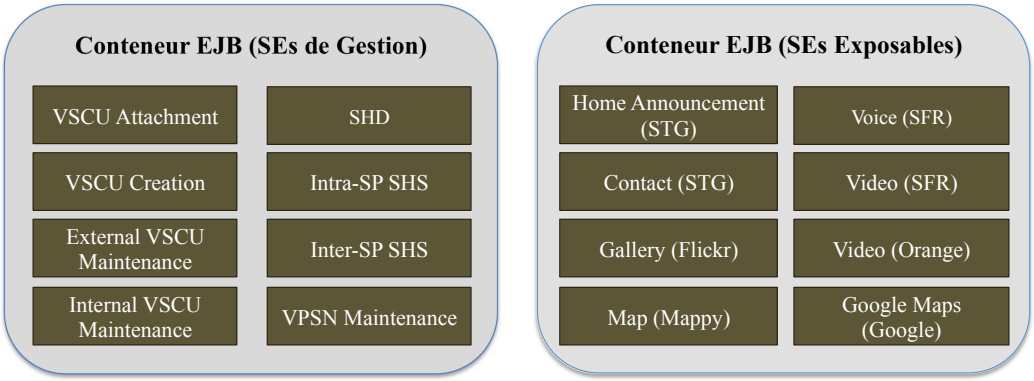

Fig. 11.13: Conteneurs EJB : services de gestion de mobilité et services exposables.

12 Conclusions et Perspectives

Cette thèse avait pour objectif de concevoir une architecture convergente pour la personnalisation et la continuité de services, dans un contexte NGN/NGS mobile et *user-centric*. Dans ce dernier chapitre, nous concluons en synthétisant l'ensemble de nos contributions (voir Section 12.1), et ensuite, nous formulons nos perspectives (voir Section 12.2).

12.1 Conclusions

Avec l'émergence du contexte NGN/NGS et de l'approche *user-centric* mobile, de nouveaux besoins régissent la relation entre les fournisseurs de services et les utilisateurs. D'après notre analyse, nous avons identifié les défis suivants : la personnalisation de services selon les besoins et les préférences de l'utilisateur, la convergence des services appartenant à différents domaines (Telco, Web et IT), l'intégration verticale entre les couches NGS et NGN tout en tenant compte des défis d'hétérogénéité et de mobilité qui régissent la convergence réseau, et enfin, la maintenance d'une session de services unique et continue, qui tient compte des préférences fonctionnelles et non-fonctionnelles (QoS demandée) de l'utilisateur, ainsi que de son contexte ambiant. Afin de lever ces différents défis, nous avons effectué une réingénierie au niveau du modèle de services et au niveau de l'architecture qui les structure. Nous avons proposé, aux niveaux architectural et fonctionnel, plusieurs solutions innovantes qui font partie d'un nouveau "écosystème" de services.

Nous avons commencé par trouver des solutions pour la convergence et la personnalisation de services, tout en tenant compte des problèmes de flexibilité des architectures verticales en silos et des problèmes de couplage fort des approches distribuées orientées objet. Pour ce faire, nous avons proposé un nouveau modèle architectural qui suit une approche évènementielle, orientée service, horizontale, distribuée et trans-organisationnelle. En effet, l'approche SOA améliorée s'appuie sur un nouveau modèle de services, qui consiste à avoir des composants de services qui sont non seulement réutilisables, interopérables et autonomes, mais aussi mutualisables, *Stateless*, autogérés à l'aide d'un agent de QoS et interconnectés par des liens génériques à couplages

lâches. De plus, ce modèle s'applique sur des éléments de services appartenant à différents domaines (Telco, Web et IT). Par conséquent, grâce à ce modèle architectural et à ce modèle de services, nous avons permis à tout utilisateur de composer d'une manière personnalisée et horizontale sa propre session de services, tout en faisant converger des services multi-domaines et trans-organisationnels. En outre, suite à notre approche EDA et grâce aux liens lâches qui relient les différents composants, la session de services créée est adaptée d'une manière dynamique et transparente à toute modification dans les préférences de l'utilisateur, à tout changement dans son contexte ambiant et à toute dégradation au niveau du fonctionnement ou de la QoS d'un des éléments de services demandés.

Afin de supporter cette nouvelle vision de services (NGS) et afin de garantir la cohabitation avec le contexte NGN sous-jacent, nous avons proposé une architecture de services, nommée NGN/NGS Middleware, qui introduit un ensemble de concept et de services structurer en trois partie : les services exposables, le SEIB et les services de base. En effet, les services exposables sont ceux exposés dans les catalogues des fournisseurs. Le SEIB est un bus d'interconnexion qui ajoute aux fonctionnalités de base d'un ESB, une nouvelle fonctionnalité d'interconnexion virtuelle. Cette dernière est soit horizontale, d'où la notion du VPSN, soit verticale, d'où la notion de l'agrégation de services. Le VPSN supporte la personnalisation de services en pré-provisionnant les éléments de services qui répondent aux préférences d'un utilisateur donné. L'agrégation de services supporte la cohabitation entre la couche service et les couches NGN sous-jacents en pré-provisionnant de bout-en-bout toutes les ressources (services, réseaux et équipements) qui vérifient les préférences d'un utilisateur donné. La dernière partie de notre NGN/NGS Middleware est celle des services de base. Ces derniers permettent d'une part de composer la logique de services demandée par l'utilisateur et d'autre part de rendre transparent le réseau NGN sous-jacent en tenant compte des défis d'hétérogénéité et de mobilité qui le régissent. Pour la gestion de l'hétérogénéité, nous avons introduit un nouveau modèle de QoS, à quatre critères (Disponibilité, Fiabilité, Capacité et Délai), qui s'applique sur toutes les ressources hétérogènes de bout-en-bout afin de les rendre homogènes. Pour la gestion de la mobilité, nous avons proposé deux solutions innovantes : les communautés virtuelles et le handover sémantique. En effet, ces deux solutions interviennent au niveau du "*Service Delivery*" afin de compléter la gestion de la mobilité qui est effectuée par les solutions existantes au niveau du "*Media Delivery*". Afin de mettre en œuvre nos solutions proposées, différents services de base de gestion ont été conçus.

Grâce à la solution des communautés virtuelles, nous avons pu maintenir la continuité de services tout en tenant compte des préférences fonctionnelles de l'utilisateur ainsi que de ses préférences non fonctionnelles, c.à.d. sa QoS demandée. Pour ce faire, nous avons proposé d'anticiper toute coupure de la session de services de l'utilisateur en remplaçant tout service dégradé par un service ubiquitaire, ayant la même fonctionnalité et une QoS équivalente. Afin de supporter cette approche, des communautés virtuelles sont créées pour combiner les services ubiquitaires, et des agents de QoS sont intégrés dans ces différents services pour détecter toute dégradation de la QoS ou du fonctionnement de chaque service. Autres que l'ubiquité, d'autres intérêts sont

aussi pris en considération afin de concevoir d'autres types de communautés virtuelles. Nous avons conçu des VSC-U&Ps qui combinent des services ubiquitaires appartenant à un même fournisseur, des VSC-U&Ls qui combinent des services ubiquitaires qui sont dans la même zone d'entourage, et enfin, nous avons combiné ces deux types de communautés pour avoir les VSCU-L&Ps. En outre, nous avons aussi favorisé dans notre approche une gestion distribuée où chaque plate-forme est responsable de ses propres communautés. Pour ce faire, nous avons différencié les notions d'Inner VSC-U&L et d'Outer VSC-U&L.

Grâce à la solution du handover sémantique, nous avons pu maintenir la continuité de services, améliorer la QoE de l'utilisateur et diminuer le temps de latence, tout en tenant compte du nouveau contexte ambiant de cet utilisateur. Pour ce faire, nous avons proposé de remplacer un service pré-provisionné dans le VPSN par un autre service ubiquitaire appartenant à la nouvelle zone ambiante de l'utilisateur. Afin de supporter cette approche, chaque fournisseur distribue ses services ubiquitaires dans des zones géographiques à grande échelle et il associe à chaque service une zone de couverture. Ainsi, lors d'une mobilité spatiale (mobilité du terminal ou mobilité de l'utilisateur), et lorsque l'utilisateur sort de la zone de couverture relative à l'un de ses services pré-provisionnés et passe vers la zone de couverture d'un nouveau service, la plate-forme du service pré-provisionné intervient pour exécuter un processus de handover sémantique. Ainsi, nous différencions deux types de handover sémantique possibles : un handover sémantique intra-fournisseur dans le cas où les deux services appartiennent au même fournisseur, et un handover sémantique inter-fournisseurs dans le cas opposé.

Au delà de ce contexte fort prometteur où l'utilisateur est au centre de l'architecture, où la composition de service répond aux besoins de personnalisation, et où nous rendons possible la convergence de services, nous pensons apporter une réponse au monde du *cloud computing* en intégrant nos solutions pour gérer les utilisateurs du *cloud*.

Ensuite, nous avons présenté un scénario qui montre l'intérêt qu'apportent nos deux solutions de gestion de la mobilité et qui vérifie la complémentarité de ces solutions. Ce scénario est mis en œuvre suite à l'intégration des services de gestion de mobilité et des services exposables dans des conteneurs EJB dans un serveur *GlassFish*. Ce dernier fait partie d'un démonstrateur qui intervient de bout-en-bout pour valider la faisabilité de nos travaux user-centric.

12.2 Perspectives

La suite de nos travaux portera au début sur l'agrégation verticale des différentes ressources de façon à adapter d'une manière dynamique les couches NGN aux modifications au niveau de la couche service. L'objectif est d'atteindre une session dynamique et continue qui tient compte des préférences de l'utilisateur, de son contexte ambiant et de sa QoS de bout-en-bout. Une des solutions possibles est la virtualisation au niveau des réseaux et des équipements. Cette virtualisation consistera à provisionner d'une manière dynamique des réseaux virtuels pour relier les différents éléments de services participant à la logique de services demandée par l'utilisateur.

En outre, nous trouvons intéressant le fait d'étudier l'influence des performances des couches réseaux sur la décision du handover sémantique. En effet, selon notre solution, le fournisseur de services décide de remplacer un service pré-provisionné dans le VPSN par un autre service ubiquitaire appartenant à la nouvelle zone ambiante. Par conséquent, nous considérons que le service qui est géographiquement le plus proche de l'utilisateur est celui qui améliore le mieux la QoE en réduisant la latence. Cependant, quand un grand nombre d'utilisateurs dans une même zone géographique pointe vers un même service, ce dernier sera normalement surchargé. Afin de dépasser ce problème, nous allons étudier une interaction possible entre notre solution et un nouveau protocole nommé ALTO (*Application Layer Traffic Optimization*). Le protocole ALTO [55][3] permet de faciliter le partage des informations réseaux avec la couche service, et peut aider les fournisseurs de services à choisir le meilleur service ubiquitaire pour remplacer le service déjà sélectionné. En effet, ALTO propose, d'après les informations réseaux temps réel, les services qui ont les niveaux de congestion réseau les moins élevés. Par conséquent, il garantit à l'utilisateur une amélioration aux niveaux des performances et de la QoE. Ce nouveau protocole peut être aussi utilisé comme un algorithme pour choisir, lors d'une dégradation d'un service pré-provisionné, le remplaçant ubiquitaire ayant le niveau de congestion réseau le plus faible parmi les différents services sélectionnés dans la communauté virtuelle correspondante.

Au niveau des études de performances, nous devrons définir les paramètres de test qui seront appliqués à notre scénario applicatif déployé, afin de démontrer l'influence positive de nos propositions de gestion de mobilité sur les performances du démonstrateur.

Enfin, puisque nos propositions se basent sur une approche trans-organisationnelle, il semble aussi intéressant de proposer des services et des modèles économiques afin de gérer les contrats de tarification et de facturation signés entre les différents fournisseurs de services.

Liste des Publications

- **Article de revue internationale avec comité de lecture**

  [R_1] Rachad Nassar, and Noëmie Simoni. Semantic handover among distributed coverage zones for an ambient continuous service session. *To appear in the International Journal of Handheld Computing Research*, 2012.

- **Articles de congrès internationaux avec comité de lecture**

  [C_1] Rachad Nassar, Soumia Kessal, and Noëmie Simoni. Semantic handover for seamless service continuity within user ambient context. *In proceedings of NGMAST*, pages 36-41, Cardiff, Wales, September 2011.

  [C_2] Rachad Nassar, and Noëmie Simoni. Cloud management architecture in NGN/NGS context : QoS-awareness, location-awareness and service personalization. *In proceedings of CLOSER*, pages 629-635, Noordwijkerhout, Netherlands, May 2011.

  [C_3] Rachad Nassar, and Noëmie Simoni. E2E mobility management in ubiquitous and ambient networks context. *In proceedings of MobiQuitous*, pages 312-323, Sydney, Australia, December 2010.

  [C_4] Rachad Nassar, and Noëmie Simoni. Composants de service pour une mobilité sans couture. *In proceedings of GRES*, Montreal, Canada, October 2010.

  [C_5] Rachad Nassar, and Noëmie Simoni. NGN/NGS components for service personalization in a mobile and heterogeneous context. *In proceedings of IWUSP*, Kyoto, Japan, October 2010.

Liste des Délivrables UBIS

[D_1] Rachad Nassar, Houda Alaoui, et Noëmie Simoni. Lot 3.3 : Spécification de la session et de la couche de service. Septembre 2011.

[D_2] Rachad Nassar, et Noëmie Simoni. Lot 3.2 : Spécifications fonctionnelles des communautés d'intérêt. Septembre 2010.

[D_3] Rachad Nassar, et Noëmie Simoni. Lot 6.1 : Architecture du démonstrateur. Juillet 2010.

[D_4] Rachad Nassar, Chunyang Yin, et Noëmie Simoni. Lot 2.1 : Définition et spécification de l'architecture. Janvier 2010.

Bibliographie

[1] ABI Research. Research report on Entreprise Mobile Cloud Computing : cloud Services, mobile devices, and the IT supply chain analysis. *http ://www.abiresearch.com/research/1004608*, 2009.

[2] Agrawal P., Yeh J.H., Chen J.C., Zhang T. IP Multimedia Subsystems in 3GPP and 3GPP2 : Overview and Scalability issues. *IEEE Communications Magazine*, 46(1) :138–145, Jan. 2008.

[3] Alimi R., Penno R., Yang Y. ALTO protocol internet draft : draft-ietf-alto-protocol-10.txt. IETF Standards Organization, Oct. 2011.

[4] Amazon. Amazon CloudFront. *http ://aws.amazon.com/cloudfront/*.

[5] Arjona A., Verkasalo H. Unlicensed Mobile Access (UMA) handover and packet data performance analysis. In *Proc. of ICDT*, pages 9–14, Barcelona, Spain, Jul. 2007.

[6] Barnatt C. Web 2.0. *http ://explainingcomputers.com/web2.html*.

[7] Barreto C., Bullard V., Erl T., Evdemon J., Jordan D., Kand K., König D., Moser S., Stout R., Ten Hove R., Trickovic I., Van Der Rijn D., Yiu A. Web Services Business Process Execution Language, version 2.0, primer. OASIS open standards consortium, May 2007.

[8] Booth D., Haas H., McCabe F., Newcomer E., Champion M., Ferris C., Orchard D. W3C working group note : web services architecture. W3C open standards community, Feb. 2004.

[9] Brenner M., Unmehopa M. *The Open Mobile Alliance : delivering service enablers for next-generation applications*, chapter Introduction, The silo syndrome and its solution, pages 1–21. John Wiley & Sons Ltd, 2008.

[10] Chinnici R., Moreau J.J., Ryman A., Weerawarana S. Web Services Description Language (WSDL), version 2.0, part 1 : CoreLanguage. W3C recommendation : WSDL. W3C open standards community, Jun. 2007.

[11] Christensen E., Curbera F., Meredith G., Weerawarana S. W3C note : Web Services Description Language (WSDL) 1.1. W3C open standards community, Mar. 2001.

[12] Clement L., Hately A., Von Riegen C., Rogers T. UDDI, version 3.0.2, UDDI spec technical committee draft. OASIS open standards consortium, Oct. 2004.

[13] Dey A.K. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1) :4–7, Feb. 2001.

[14] Distefano S., Puliafito A., Rak M., Venticinque S., Villano U., Cuomo A., Di Modica G., Tomarchio O. QoS management in Cloud@Home infrastructures. In *Proc. of CyberC*, pages 190–197, Beijing, China, Oct. 2011.

[15] Dowlatshahi M., Safaei F. Mobility management for untethered immersive communications. In *Proc. of ICME*, pages 1213–1216, Toronto, Canada, Jul. 2006.

[16] Erl T. *Service-Oriented Architecture : concepts, technology, and design*, chapter SOA and Service-Orientation, pages 244–313. Prentice Hall PTR, Aug. 2005.

[17] ETSI. GSM TS 03.09, version 5.1.0 : Digital cellular telecommunications system (phase 2+) ; Handover Procedures. GSM Technical Specification, Aug. 1997.

[18] ETSI. RRC Connection Mobility. In *3GPP TR 25.922 version 7.1.0, release 7 : Universal Mobile Telecommunications System (UMTS) ; Radio resource management strategies*, 3GPP Technical Report, pages 15–41, Mar. 2007.

[19] ETSI. 3GPP TS 23.009, version 9.0.0, release 9 : digital cellular telecommunications system (Phase 2+) ; Universal Mobile Telecommunications System (UMTS) ; Handover procedures. 3GPP Technical Specification, Jan. 2010.

[20] ETSI. 3GPP TS 43.129 version 10.0.0, release 10 : Digital cellular telecommunications system (phase 2+) ; Packed-switched handover for GERAN A/Gb mode ; stage 2. 3GPP Technical Specification, Apr. 2011.

[21] ETSI. 3GPP TS 43.318 version 10.1.0, release 10 : Digital cellular telecommunications system (Phase 2+) ; Generic Access Network (GAN) ; stage 2. 3GPP Technical Specification, Apr. 2011.

[22] ETSI. Mobility. In *3GPP TS 36.300, version 9.8.0, release 9 : LTE ; Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN) ; overall description ; stage 2*, 3GPP Technical Specification, pages 44–75, Nov. 2011.

[23] ETSI. Mobility Management Functionality. In *3GPP TS 23.060, version 10.3.0, release 10 : Digital cellular telecommunications system (phase 2+) ; Universal Mobile Telecommunications System (UMTS) ; General Packet Radio Service (GPRS) ; Service description ; stage 2*, 3GPP Technical Specification, pages 55–179, Mar. 2011.

[24] Fielding R.T. *Architectural styles and the design of network-based software architectures.* Dissertation in Information and Computer Science, University of California, Irvine, 2000.

[25] FOKUS. Open IMS Core. *www.open-ims.org/core/.*

[26] Foster I., Zhao Y., Raicu I., Lu S. Cloud Computing and Grid Computing 360-degree compared. In *Proc. of GCE*, pages 1–10, Texas, USA, Nov. 2008.

[27] Fournier-Morel X., Grojean P., Plouin G., Rognon C. *SOA : Le guide de l'architecte d'un SI agile, 3^{me} édition*, chapter Modéliser les services, pages 89–111. Dunod, 2011.

[28] Garrahan J.J., Russo P.A., Kitami K., Kung R. Intelligent Network overview. *IEEE Communications Magazine*, 31(3) :30–36, Mar. 1993.

[29] Giaffreda R., Karmouch A., Jonsson A., Karlsson A.M., Smirnov M.I., Glitho R., Galis A. Context-aware communications in ambient networks. In *Proc. of WWRF*, Oslo, Norway, Jun. 2004.

[30] Gopalan K.S., Nathan S., Teja C.H.B., Channa A.B., Saraf P., Shanker G. A cloud based service architecture for personalized media recommendations. In *Proc. of NGMAST*, pages 19–24, Cardiff, Wales, Sep. 2011.

[31] Gouya A., Crespi N., Bertin E. Service invocation issues within the IP Multimedia Subsystem. In *Proc. of ICNS*, pages 33–33, Athens, Greece, Jun. 2007.

[32] Parlay X Working Group. Parlay APIs 4.0, version 1.0 : Parlay X web services white paper. Parlay Group, May 2003.

[33] Gudgin M., Hadley M., Mendelsohn N., Moreau J.J., Nielsen H.F., Karmarkar A., Lafon Y. SOAP, version 1.2, part 1 : messaging framework, second edition. W3C Recommendation : SOAP. W3C open standards community, Apr. 2007.

[34] Guo H., Chen J., Wu W., Wang W. Personalization as a Service : the architecture and a case study. In *Proc. of CloudDB*, pages 1–8, HongKong, China, Nov. 2009.

[35] Heine G. *GSM networks : protocols, terminology, and implementation*, chapter Handover, pages 251–256. Artech House Mobile Communications series, 1999.

[36] HP Laboratories. Datasheet of the HP Business Availability Center software : Improving IT operational efficiency and customer satisfaction. Technical report, May 2009.

[37] IBM. IBM introduces ready-to-use cloud computing : "Blue Cloud". *http ://www-03.ibm.com/press/us/en/pressrelease/22613.wss*, Nov. 2007.

[38] IDC market intelligence and advisory firm. IT cloud services user survey, pt.2 : Top benefits and challenges. *http ://blogs.idc.com/ie/ ?p=210*, Oct. 2008.

[39] ISO. ISO 20000-1 IT Service management standard : specification for service management. 2005.

[40] ITU-T. CS-1 architecture, infrastructure in CS-1. In *ITU-T recommandation Q.1219 : Intelligent Network user's guide for capability set 1*, pages 14–37, Apr. 1994.

[41] ITU-T. ITU-T recommendation Y.2001 : general overview of NGN. Y-Series recommendations : global information infrastructure, internet protocol aspects and next-generation networks, next generation networks - frameworks and functional architecture models, Dec. 2004.

[42] Kavantzas N., Burdett D., Ritzinger G., Fletcher T., Lafon Y., Barreto C. Web Services Choreography Description Language, version 1.0. W3C candidate recommendation : WS-CDL. W3C open standards community, Nov. 2005.

[43] Knightson K., Morita N., Towle T. NGN architecture : generic principles, functional architecture, and implementation. *IEEE Communications Magazine*, 43(9) :49–56, Oct. 2005.

[44] Korn A., Peltz C., Mowbray M. A Service Level Management Authority in the cloud. Technical report, HP Laboratories, Apr. 2009.

[45] Mathieu B., Simoni N. *Des réseaux intelligents à la nouvelle génération de services*, chapter Réseaux overlays de services pour les réseaux ambiants, pages 267–269. Hermès Science Publications, Lavoisier, Feb. 2007.

[46] Michelson B.M. Event-driven architecture overview, event-driven soa is just part of the eda story. Technical report.

[47] Moerdijk A.J., Klostermann L. Opening the networks with Parlay/OSA : standards and aspects behind the APIs. *IEEE Network*, 17(3) :58–64, May 2003.

[48] OMA. OMA Service Environment, approved version 1.0.4. Feb. 2007.

[49] Oracle. Database 11g. *http ://www.oracle.com/technetwork/database/enterprise-edition/documentation/index.html*.

[50] Perkins C.E. Mobile networking through Mobile IP. *IEEE Internet Computing*, 2(1) :58–69, Jan.-Feb. 1998.

[51] Podhradsky P. Migration scenarios and convergence processes towards NGN (present state and future trends). In *Proc. of ELMAR*, pages 39–46, Zadar, Croatia, Jun. 2004.

[52] Rigault C., Simoni N. *Des réseaux intelligents à la nouvelle génération de services*, chapter Le réseau intelligent et les prémises des technologies de services, pages 29–74. Hermès Science Publications, Lavoisier, Feb. 2007.

[53] Rostambeik S., Simoni N. *Des réseaux intelligents à la nouvelle génération de services*, chapter Les architectures de services : supports d'ouverture, Parlay, pages 133–137. Hermès Science Publications, Lavoisier, Feb. 2007.

[54] Saboowala H. Cloud service placement : selecting the optimal data center location for the cloud consumer. *Cloudbook*, 2(1) :13–16, 2011.

[55] Seedorf J., Kiesel S., Stiemerling M. Traffic localization for P2P-applications : the ALTO approach. In *Proc. of P2P*, pages 171–177, Seattle, U.S.A., Sep. 2009.

[56] Shroff G. *Enterprise cloud computing : technology, architecture, applications*, chapter Web services, AJAX and mashups, pages 77–88. Cambridge University Press, Dec. 2010.

[57] Simoni N., Yin C., Du Chene G. An intelligent user centric middleware for NGN : Infosphere and AmbientGrid. In *Proc. of COMSWARE*, pages 599–606, Bangalore, India, Jan. 2008.

[58] IEEE Computer Society. MIHF services. In *IEEE Std 802.21-2008 : part 21 : Media Independent Handover Services*, IEEE standard for local and metropolitan area networks, pages 35–59. IEEE standards association, Jan. 2008.

[59] IEEE Computer Society, IEEE Microwave Theory, and Techniques Society. Macro diversity handover (MDHO) and fast BS switching. In *IEEE Std 802.16-2009 : part 16 : Air interface for broadband wireless access systems*, IEEE standard for local and metropolitan area networks, pages 463–480. IEEE standards association, May 2009.

[60] Sringanesh R.P., Brose G., Silverman M. *Mastering Entreprise JavaBeans 3.0.* Wiley Publishing Inc., 2006.

[61] SUN Microsystems. GlassFish Server. *http ://glassfish.java.net/.*

[62] SUN Microsystems. JXTA. *https ://jxta.dev.java.net/.*

[63] SUN Microsystems. Sailfin. *http ://sailfin.java.net/.*

[64] Vaughan-Nichols S.J. Will mobile computing's future be location, location, location ? *IEEE Computer Journal*, 42(2) :14–17, Feb. 2009.

[65] VirtuOR. Réseaux Virtuels VirtuOR. *http ://www.virtuor.fr/.*

[66] Vollmer K., Gilpin M., Rose S. The Forrester Wave^{TM} : Enterprise Service Bus, Q2 2011. Technical report, Forrester research and advisory firm, Apr. 2011.

[67] Wang Z., Yang F. A multiple-mode mobile location-based information retieve system. In *Proc. of ICWMC*, pages 410–415, Cannes, France, Aug. 2009.

[68] Xie J., Howitt I., Shibeika I. IEEE 802.11-based mobile IP fast handoff latency analysis. In *Proc. of ICC*, pages 6055–6060, Glasgow, Scotland, Jun. 2007.