Ministère de l'enseignement supérieur et de la recherche scientifique

UNIVERSTE SAAD DAHLEB DE BLIDA 1

FACULTE DES SCIENCES

DEPARTEMENT D'INFORMATIQUE

Mémoire de Fin d'Etude de

Master en IngénierieDu Logiciel

Thème

Modélisation sémantique pour l'interopérabilité entre Clouds

Domaine : MI

Filière : Informatique

Spécialité : Ingénierie du logiciel

Encadré par :

Mme MANCER Yasmine

Rédigé par :

Melle SLIMANI Yasmina

Melle MESSABIH Hala

Année Universitaire : 2016/2017

Président de jury :
Fahria Fatma Zohra

Examinateur du Mémoire :
Guessoum Dalila

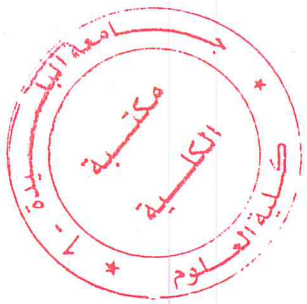
Résumé

Dans l'environnement du Cloud Computing, l'interopérabilité est l'un des obstacles qui retarde l'adoption de systèmes logiciels en tant que service. Le problème est que les logiciels Cloud actuels en tant que service n'ont pas été construits au départ avec des mécanismes d'interopérabilité. En général, l'interopérabilité fait référence à la capacité de communication entre fournisseurs Cloud à travers l'échange de services.

Le présent mémoire introduit une solution pour l'interopérabilité sémantique des services Cloud de type logiciel en tant que service (SaaS). La solution proposée consiste à utiliser la planification de l'intelligence artificielle à travers l'algorithme Graphplan. Le but étant de trouver un plan adéquat pour réaliser la communication entre Clouds.

Pour l'expérimentation de la solution proposée, nous avons utilisé le simulateur CloudSim. CloudSim est actuellement le simulateur le plus utilisé dans le domaine du Cloud Computing. Les tests effectués ont donné de bons résultats surtout en termes de temps d'interopérabilité.

Mots clés : Cloud Computing, SaaS, interopérabilité sémantique, Graphplan, CloudSim.



Abstract

In the Cloud Computing environment, interoperability is one of the barriers that delay the adoption of software systems as a service. The problem is that the current Cloud software as a service was not originally built with interoperability mechanisms. In general, interoperability refers to the ability of communication between Cloud provides through the exchange of services.

The present paper introduces a solution for the semantic interoperability of software services as a service (SaaS). The proposed solution consists in using the planning of artificial intelligence through the Graphplan algorithm. The goal is to find a suitable plan to achieve communication between Clouds.

For the experimentation of the proposed solution, we used the CloudSim simulator. CloudSim is currently the most widely used simulator in the Cloud Computing domain. The tests performed gave good results especially in terms of interoperability time.

Key words: Cloud Computing, SaaS, semantic interoperability, Graphplan, CloudSim.

الملخص

في بيئة الحوسبة السحابية، تعد قابلية التشغيل البيني واحدة من الحواجز التي تؤخر اعتماد أنظمة البرمجيات كخدمة. والمشكلة هي أن البرنامج السحابي الحالي كخدمة لم يكن مبنيا في الأصل مع آليات التشغيل البيني. وبوجه عام، تشير قابلية التشغيل البيني إلى قدرة الاتصال بين مقدمي الخدمات السحابية من خلال تبادل الخدمات.

تقدم هذه الأطروحة حلا للتشغيل البيني الدلالي لخدمات البرمجيات كخدمة (SaaS). الحل المقترح يتمثل في استخدام تخطيط الذكاء الاصطناعي من خلال خوارزمية غرافبلان. الهدف يكمن في إيجاد خطة مناسبة لتحقيق التواصل بين الغيوم.

لتجربة الحل المقترح، استخدمنا محاكاة كلودسيم. كلودسيم تعتبر حاليا المحاكاة الأكثر استخداما في مجال الحوسبة السحابية. أعطت الاختبارات التي أجريت نتائج جيدة وخاصة من حيث توقيت التشغيل البيني.

الكلمات المفتاحية : الحوسبة السحابية, SaaS, قابلية التشغيل البيني الدلالي, غرافبلان, كلودسيم.

Remerciement

Tout d'abord, nous remercions Allah, le tout puissant, qui nous a donné la force, la patience et la volonté pour accomplir ce modeste travail.

Nous tenons à exprimer notre gratitude à notre promotrice ***Yasmine Mancer*** *pour son encadrement, sa disponibilité, son suivi, ses conseils précieux et son encouragement.*

Nous remercions les membres du jury qui nous ont fait l'honneur d'accepter de juger notre travail.

Nous tenons à remercier également Mr ***Volker Strobel*** *professeur à l'université libre de Bruxelles pour son aide et sa disponibilité.*

Un grand MERCI à nos familles et à nos amies qui étaient notre soutien moral.

SOMMAIRE

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ACRONYMES ET ABRÉVIATIONS

AOP	Aspect Oriented Program-Ming
API	Application Programming Interface
ASP	Agence de Services et de Paiement
BW	Band With
CRM	Customer Relationship Management
CaaS	Connection as a Service
COaaS	Communication as a Service
CIMI	Cloud Infrastructure Management Interface
CAMP	Cloud Application Management for Platforms
CDMI	Centre de Distribution de Matériel Informatique
CAML	Categorical Abstract Machine Language
CRUD	Create, Retrieve, Update and Delete
C3	Chrysler Comprehensive Compensation System
CIS	Cloud Information Services
CPU	Central Processing Unit
CDDL	Common Development and Distribution License
DaaS	Data as a Service

DAG	Graphique Acyclique Dirigé
DSM	DiskStation Manager
DSL	Digital Subscriber Line
EC2	Amazon Elastic Compute Cloud
EIF	European Investment Fund
ENISA	European Network and Information Security Agency
EMML	Enterprise Mashup Markup Language
ECML	Elastic Computing Modeling Language
EDML	Elastic Deployment Modeling Language
EMLM	Elastic Manage-Langage de modélisation
ebXML	Electronic Business XML Initiative
FIPA-ACL	FIPA1 Agent Communication Langage
FOAF	Friend-Of-A-Friend
HTTP	HyperText Transfer Protocol
HTML	HyperText Markup Language
IaaS	Infrastructure as a Service
IBM	International Business Machines
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
IC	Inter-Cloud

ICA	InterCloud Architecture
IDE	Investissements Directs à l'Etranger
JSON	JavaScript Object Notation
JSP	JavaServer Pages
JRE	Java Runtime Environment
JVM	Java Virtual Machine
KQML	Knowledge Query and Manipulation Language
MDA	Model Driven Architecture
MAS	Multi-Agents System
NaaS	Network as a Service
NIST	National Institute of Standards and Technology
NASA	National Aeronautics and Space Administration
Open-CSA	Open Composite Services Architecture
OV	Organisation Virtuelle
OMG	Object Management Group
OCCI	Open Cloud Computing Interface
OVF	Format Open Virtualization
OWL	Web Ontology Language
OWL-S	Sematic Web Ontology Language
PaaS	Platform as a Service

PME	Petite et Moyenne Entreprise
PDDL	Planning Domain Definition Language
PHP	Hypertext Preprocessor
RASIC	Responsible, Approves, Supports, Informed, Consulted
RDF	Resource Description Framework
RDFS	Resource Description Framework Semantic
SaaS	Software as a Service
SEaaS	Security as a Service
SLA	Service Level Agreement
SI	Système d'Information
SOAP	Simple Object Access Protocol
SCA	Service Component Architecture
SOA	Service Oriented Architecture
SWRL	Semantic Web Rule Language
TOSCA	Topology and Orchestration Specification for Cloud Application
USDL	Unified Service Description Language
UML	Unified Modeling Language
USDL	Unified Service Description Language
UE	Union Européenne

VMDK	Virtual Machine Disk
VEEM	Virtual Execution Environment Manager
VM	Virtual Machine
WSDL	Web Services Description Language
XML	eXtensible Mark-up Language
XPDL	XML Process Definition Language
XP	eXtreme Programming

Introduction Générale

Malgré le fort engouement que suscite le Cloud Computing, son adoption et sa vulgarisation se heurtent à plusieurs obstacles tels que : l'hétérogénéité des architectures et des API, le grand nombre de fournisseurs et de technologies, l'absence de standard, le problème du Vendor Lock-In[1], etc.

Une des solutions pour faire face à ces problèmes, c'est d'aller vers des Clouds ouverts. Dans ce type de Cloud, l'utilisateur a la liberté de choisir entre les différents fournisseurs et les services Cloud, selon son besoin qui peut changer au cours du temps. Cependant, pour parvenir à l'ouverture du Cloud, il faut d'abord régler les problèmes d'interopérabilité.

L'Inter-Cloud Computing est un modèle de Cloud qui vise à garantir la qualité de service de bout en bout (disponibilité et performance), en permettant une réaffectation à la demande des ressources et un transfert de la charge de travail. Cela, grâce à une interopérabilité des systèmes de Clouds de différents fournisseurs. L'interopérabilité se base sur la coordination des exigences de qualité de service de chaque consommateur avec le SLA (Service Level Agreements) de chaque fournisseur et l'utilisation d'interfaces.

Si un Cloud subit une surcharge inattendue, il aura besoin de ressources pour faire face à cette situation. Cependant, étant donné que le nombre de ressources disponibles pour un seul Cloud est limité, le système peut ne pas être en mesure de fournir les services demandés. Afin de garantir la qualité du service requis, en termes de disponibilité et de performance, même dans de tels cas, il sera nécessaire de fournir

[1] Situation d'enfermement propriétaire.

un mécanisme pour réaffecter les ressources de manière flexible entre les Clouds. Dans ce cas, il s'agit de fédération horizontale de Clouds.

En général, l'interopérabilité est définie comme la capacité des systèmes et les processus métiers qu'ils supportent à échanger des données et à permettre la distribution d'informations et de connaissances [1]. L'interopérabilité des logiciels en tant que service dans l'environnement du Cloud Computing se rapporte à la capacité de deux logiciels différents en tant que service à coopérer ou à interagir entre eux [2]. Par conséquent, l'interopérabilité est une condition préalable à la coopération entre ce type de services. Cela permet aux clients la flexibilité d'exécuter des systèmes localement, dans le Cloud, ou dans une combinaison des deux Clouds.

a. <u>Problématique</u> :

Le manque de standardisation dans le domaine du Cloud Computing a fait émerger beaucoup de problèmes relatifs au nombre grandissant de fournisseurs de services Cloud d'un côté, et d'un autre côté, à l'hétérogénéité des technologies et plateformes utilisées. Delà, un réel besoin de collaboration, et donc d'interaction, est ressenti par les différents intervenants dans le Cloud. Ce besoin a donné naissance à l'Inter-Cloud Computing.

À l'heure actuelle, les modèles et les solutions d'interopérabilité existants pour les logiciels en tant que service dans l'environnement du Cloud Computing ne sont toujours pas satisfaisants car ils ne peuvent couvrir que l'interopérabilité syntaxique et ils ne sont pas capables de fournir une interopérabilité sémantique, et cela dû au manque de modèles et de méthodologies répondant correctement aux défis d'interopérabilité. Par conséquent, fournir une interopérabilité sémantique pour les logiciels en tant que service dans l'environnement de Cloud Computing est une préoccupation majeure.

La problématique de ce travail consiste à définir un modèle pour l'interopérabilité au niveau sémantique entre les services Cloud de type SaaS.

b. Objectifs :

Les objectifs de notre travail sont les suivants :

1. Etude comparative des solutions existantes pour l'interopérabilité sémantique entre Clouds.
2. Étudier et analyser les exigences d'interopérabilité sémantique dans l'environnement du Cloud Computing.
3. Définir un modèle pour l'interopérabilité au niveau sémantique entre Clouds de type SaaS. Cette définition doit prendre en compte les aspects suivants :
    a. Description des interfaces de communication des services.
    b. Description des règles d'interaction entre les services.
4. Validation de la solution proposée par une expérimentation en utilisant le simulateur Cloudsim.

c. Organisation du mémoire :

Le présent mémoire est organisé de la manière suivante :

Chapitre I : présente un aperçu sur les concepts de base du Cloud Computing, y compris les définitions, les modèles de déploiement, les modèles de service, les caractéristiques essentielles, les obstacles et les avantages du Cloud ainsi que l'évolution du Cloud Computing vers l'inter-Cloud.

Chapitre II : consacré à l'étude des notions de bases de l'interopérabilité entre SI, et ses différentes facettes.

Chapitre III : introduit une étude détaillée de l'interopérabilité dans le Cloud Computing. Nous avons commencé par l'évolution de l'interopérabilité ensuite nous avons présenté ses deux dimensions et nous avons fini par une étude comparative des standards et des solutions existants réalisant l'interopérabilité sémantique dans l'environnement du Cloud Computing.

Chapitre IV : présente la conception de la solution proposée réalisant l'interopérabilité sémantique des services Cloud de type logiciels en tant que service. Nous avons

expliqué notre approche basée sur la planification en utilisant un des algorithmes de l'IA afin d'assurer une planification valide des interactions de services concernées.

Chapitre V : décrit les critères d'évaluation et l'expérimentation de la solution proposée pour l'interopérabilité sémantique des logiciels en tant que service dans l'environnement du Cloud Computing en utilisant le simulateur Cloudsim.

Chapitre I. *Cloud Computing*

I.1. Introduction :

Longtemps avant que l'expression "***Cloud Computing***" ne naisse, les architectes de réseaux (ceux qui conçoivent les réseaux intra et inter-entreprises) schématisaient internet par un nuage dans leurs croquis. En anglais, on parlait alors de "The Cloud", ce qui signifiait à peu près l'internet que nous connaissons. Ce nuage évoquait alors une connexion vers une quantité indéfinie d'utilisateurs et non pas des services tel que nous l'entendons maintenant.

L'informatique dans le nuage (en anglais, *Cloud Computing*) est devenu un concept majeur faisant référence à l'utilisation de la mémoire, des capacités de calcul, des ordinateurs et des serveurs répartis dans le monde entier et liés par un réseau, tel internet.

Les utilisateurs (le plus souvent des entreprises, mais aussi de simples clients) ne sont plus propriétaires de leurs serveurs informatiques mais peuvent ainsi accéder de manière évolutive à de nombreux services en ligne sans avoir à gérer l'infrastructure sous-jacente, souvent complexe. Les applications et les données ne se trouvent plus sur l'ordinateur local, mais - métaphoriquement parlant - dans un Cloud composé d'un certain nombre de serveurs distants interconnectés au moyen d'une excellente bande passante indispensable à la fluidité du système. L'accès au service se fait par une application standard facilement disponible, la plupart du temps un navigateur web.

Le but de ce chapitre est de se familiariser avec les notions de base du Cloud Computing (caractéristiques, types de déploiement, types de services, etc.) et de voir à la fin son évolution vers l'inter-Cloud.

I.2. Définition du Cloud Computing :

Le concept du Cloud Computing est comparable à celui de la distribution de l'énergie électrique. La puissance de calcul et de stockage de l'information est proposée à la consommation par des compagnies spécialisées. De ce fait, les entreprises n'ont plus besoin de serveurs propres, mais confient cette ressource à une entreprise qui leur garantit une puissance de calcul et de stockage à la demande.

Un Cloud est un système de traitement parallèle distribué qui est constitué d'un ensemble interconnecté d'ordinateur virtuel qui donne une ou plusieurs ressources informatiques unifiées et basées sur les exigences entre les fournisseurs et les consommateurs de services [3].

Le Cloud Computing ep st un nuage de services et de données [4]. Le NIST [5] a défini un Cloud ainsi : « Le Cloud Computing est un modèle pour permettre, un accès rapide, à la demande, à un ensemble partagé de ressources informatiques configurables (exemple : réseau, les serveurs, stockage, application et service) qui peuvent être rapidement approvisionnées et publiées avec un effort minimal de gestion ou d'interaction du prestataire de service».

Le Cloud Computing est une nouvelle façon de délivrer les ressources informatiques, et non une nouvelle technologie, il peut également être vu comme un réseau informatique distribué, dont les ressources peuvent être approvisionnées de manière dynamique et reposant sur un contrat de service entre un fournisseur et un client.

I.3. Caractéristiques du Cloud Computing :

Généralement, un service, une solution ou un environnement d'exécution devrait satisfaire une liste de caractéristiques pour qu'il soit considéré comme étant du Cloud Computing. Parmi ces caractéristiques, il y a celles qui sont reconnues comme fondamentales. Par exemple, le NIST définit cinq caractéristiques essentielles qui sont [5,6,7] :

I.3.3. Ressources à la demande :

Un utilisateur peut allouer unilatéralement des ressources informatiques (serveurs, réseau, stockage, environnement d'exécution, application) au besoin, de façon automatique et sans nécessité d'interaction humaine avec chaque fournisseur de service.

I.3.2. Large accès réseau :

Les ressources Cloud Computing sont disponibles à travers le réseau et accessibles via des mécanismes standards qui favorisent leurs utilisations à partir des appareils clients hétérogènes, voire légères (ex ordinateurs portables, téléphones, tablettes).

I.3.3. Mutualisation des ressources :

Les ressources informatiques du fournisseur Cloud Computing sont mutualisées pour servir plusieurs clients en utilisant un modèle multi-tenant. Ces ressources, physiques ou virtuelles, sont allouées et libérées dynamiquement selon la demande du consommateur. Généralement, l'utilisateur n'a ni le contrôle ni la connaissance de l'emplacement exact des ressources allouées. Dans certains cas, il peut choisir l'emplacement géographique à un niveau haut (ex : par pays, continent ou Datacenter).

I.3.4. Élasticité rapide :

Les ressources sont allouées et libérées d'une façon élastique, idéalement d'une façon automatique, pour s'adapter rapidement à la demande qu'elle soit croissante ou décroissante. Pour le consommateur, les ressources disponibles à l'allocation apparaissent comme illimitées et peuvent s'allouer à tout moment.

I.3.5. Services mesurés :

Toutes les ressources allouées peuvent être surveillées et contrôlées afin de mesurer leurs consommations avec un niveau d'abstraction approprié selon le type du service (ex : stockage, temps de calcul, bande passante). De plus, cela garantit un niveau de disponibilité adapté aux besoins spécifiques des clients [8].

I.4. Modèles de déploiement du Cloud Computing :

Quel que soit le modèle de service utilisé (SaaS, PaaS ou IaaS), le NIST a défini trois modèles de déploiement pour les services de Cloud, avec des variantes qui répondent à des besoins spécifiques :

I.4.1. Cloud privé :

L'infrastructure du Cloud Computing est configurée pour une utilisation exclusive par une organisation unique, un tiers, ou une combinaison des deux, et il peut exister sur ou à l'extérieur des locaux [5].

Le Cloud privé peut aussi désigner un Cloud déployé sur une infrastructure physique dédiée et mise à disposition d'un fournisseur de services. Ainsi une entreprise peut louer à un fournisseur de services, un nombre conséquent de serveurs qui lui sont entièrement dédiés et sur lesquels une solution de Cloud sera déployée pour gérer dynamiquement l'application, la plate-forme ou l'infrastructure (virtuelle) [4].

I.4.2. Cloud communautaire :

L'infrastructure de Cloud Computing est configurée pour une utilisation exclusive par une communauté de consommateurs ou des organisations qui ont des préoccupations communes (par exemple : la mission et les exigences en matière de sécurité). Il peut être administré, géré et exploité par un ou plusieurs des organismes de la communauté, un tiers, ou une combinaison des deux, et il peut exister sur ou à l'extérieur des locaux [5].

I.4.3. Cloud public :

L'infrastructure de Cloud Computing est configurée pour ouvrir l'usage au public. Il peut être administré, géré et exploité par une entreprise, université, organisation gouvernementale ou une combinaison entre eux. Il existe sur les locaux du fournisseur de services Cloud [5].

Les fournisseurs de Cloud public facturent à l'utilisation et garantissent une disponibilité de services à travers des contrats SLA (document qui définit la qualité de service requise entre un prestataire et un client) [4].

I.4.4. Cloud hybride :

L'infrastructure de Cloud Computing est une composition de deux ou plusieurs des infrastructures de Cloud Computing (privé, communautaire ou public) qui restent des entités uniques, mais qu'ils sont liées par la technologie, normalisées ou propriétaires, ce qui permet la portabilité des données et des applications (par exemple, l'éclatement des Clouds pour l'équilibrage de charge entre eux) [5].

Nous pouvons ainsi déporter les applications vers un Cloud public qui consommera des données stockées et exposées dans un Cloud privé, ou bien faire communiquer deux applications hébergées dans deux Clouds privés distincts, ou encore consommer plusieurs services hébergés dans des Cloud publics différents. Dans tous ces cas de figure, nous avons affaire à la notion de Cloud hybride [4].

I.5. Types de services :

I.5.1. Infrastructure as a Service (IaaS) :

Dans ce modèle, le client dispose d'une infrastructure informatique hébergée sur laquelle il a un accès complet (sans restriction). A la différence des services traditionnels, l'infrastructure mise au service du client n'est plus une infrastructure physique (un parc de serveurs) mais une infrastructure virtuelle [5].

I.5.2. Platform as a Service (PaaS) :

Le fournisseur met à disposition des clients un environnement fonctionnel et performant. Le client n'aura plus qu'à déployer son application [5].

I.5.3. Software as a Service (SaaS) :

Il s'agit de la mise à disposition d'un logiciel non pas sous la forme d'un produit que le client installe en interne sur ses serveurs, mais en tant qu'application accessible à distance comme un service, par le biais d'internet. Ce modèle permet de déporter l'application chez un tiers [9].

I.5.4. X as a Service (XaaS) :

Le style de nomenclature ' X as a service' a été utilisé pour caractériser des services et ressources Cloud Computing. Chacune des nouvelles abréviations peut être vues comme un sous-ensemble d'un ou plusieurs des trois types de base. Parmi ces abréviations, nous citons quelque unes d'entre elles dans le (tableau 1) suivant :

Abréviation	Désignation	Exemple d'utilisation
DaaS	Data as a Service	DaaS associe à une partie de l'entreprise **CRM (Customer Relationship Management)** en temps réel des données pour un meilleur ciblage du flux de consommateurs dans les marchés [10].
NaaS	Network as a Service	**Cisco IBSG** a créé une nouvelle architecture de solution pour les fournisseurs de services réseau mobile en tant que service [11].
CaaS	Connexion as a Service	**Amazon Elastic Compute Cloud (EC2)** a réalisé la connexion par le serveur blueprint designer qui se connecte au service d'identité Keystone et éventuellement, à un serveur LDAP [12].
COaaS	Communication as a Service	**Véronique Dompé Valette**, directrice marketing pour la vidéo conférence chez **Orange Business Services**, a utilisé CaaS pour économiser sur les voyages, ainsi que pour accélérer le processus de prise de décision [13].
SEaaS	Security as a Service	**McAfee Security SaaS - McAfee** propose un certain nombre de services externalisés tels que le nœud final, e-mail, Web et réseau protection [14].

Tableau 1 : Les abréviations à base de 'XaaS'

La (figure 1) présente les couches de base de prestations de services dans le Cloud Computing.

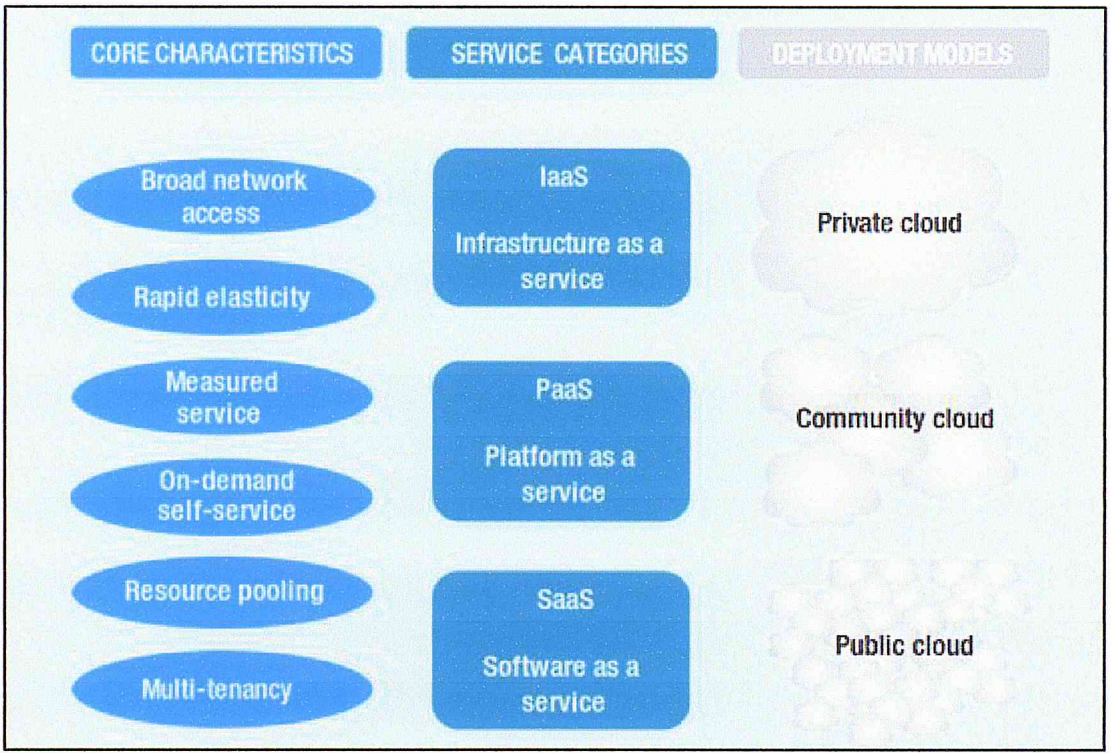


Figure 1 : Couches de prestation des services dans le Cloud [15]

I.6. Avantages et inconvénients du Cloud Computing :

I.6.1. Les avantages du Cloud Computing :

Le Cloud Computing offre de nombreux avantages que plusieurs travaux ont listés [16, 17, 18, 19, 20, 21,22]. Parmi ces avantages, il y a :

- Réduction des coûts des infrastructures ;
- Réduction des coûts de développement ;
- Réduction des coûts des logiciels ;
- Des ressources et services plus rapides à allouer et plus simples à utiliser ;
- Accès aux ressources plus flexible ;
- Meilleure utilisation plus efficace des ressources ;
- Augmentation de la puissance de calcul ;
- Grande capacité de stockage (quasi illimitée) ;

- Moins de problèmes d'entretien ;
- Gestion des mises à jour plus simple et rapide ;
- Pas de perte de données ;
- Tout est considéré comme un service défini par un SLA ;
- Infrastructure allouée et disponible juste à temps ;
- Réduction du temps de mise sur le marché ;
- Garantie d'intégrité et de disponibilité des Datacenter des opérateurs Cloud ;
- Fin du syndrome de l'« administrateur héroïque » ;
- Mashups[2] ;
- Plus d'agilité pour les etudes ;
- Plus d'agilité pour la production.

I.6.2. Les inconvénients du Cloud Computing :

Le Cloud Computing n'a pas que des avantages, il souffre aussi d'un certain nombre d'obstacles qui sont abordés dans [22,23, 19,24, 16]. Parmi ces obstacles, il y a :

- Données Lock-in[3] ;
- Confidentialité des données ;
- Chiffrement des données ;
- Nécessité d'un accès réseau constant ;
- Mauvais fonctionnement avec les connexions à basse vitesse ;
- Faible niveau de la qualité de service dans le réseau ;
- Risque d'engorgements lors des transferts de données ;
- Problème d'interopérabilité ;
- Problème de portabilité ;
- Faible contrôlabilité ;
- Manque de fonctionnalités d'audit ;
- Des contrats de service SLAs non normalisés ;

[2] C'est une application ou widget qui permet sur un site web d'agréger ou retraiter de l'information en provenance d'une ou plusieurs sources extérieures.

[3] Signifie qu'une technologie particulière n'est pas choisie parce qu'elle est la meilleure mais parce qu'elle a été précédemment choisie dans le marché.

- Les problématiques juridiques et commerciales ;
- Manque de sécurité.

Malgré ses inconvénients, le cloud computing est devenu incontournable dans la mise en place et la fourniture de services informatiques.

I.7. Evolution du Cloud Computing :

I.7.1. Historique de l'évolution du Cloud Computing :

Le concept du Cloud Computing consiste à fournir des ressources informatiques sous forme de services dans lesquels l'utilisateur paie pour ce qu'il utilise.

Ce concept est apparu dans les années 60 notamment avec McCarthy [25] ou encore Kleinrock [26] sous le nom d'informatique utilitaire. C'est ensuite vers la fin des années 90 que ce concept a réellement pris de l'importance avec tout d'abord le Grid Computing [27], ce terme est une métaphore exprimant la similarité avec le réseau électrique dans lequel l'électricité est produite dans de grandes centrales et il est disséminée à travers un réseau jusqu'aux utilisateurs finaux. Les grandes centrales correspond au Datacenter, le réseau est le plus souvent celui d'internet et l'électricité correspond aux ressources informatiques.

Le terme *Cloud Computing* n'est véritablement apparu qu'au cours des années 2006-2008 [28] avec l'apparition d'amazon EC2 [29] ou encore la collaboration d'IBM et Google [30,31] ainsi que l'annonce d'IBM concernant 'Blue Cloud' [32]. Par la suite de nombreuses solutions open source ont aussi vu le jour avec par exemple OpenShift[4] [33] de RedHat, ou encore OpenStack[5] [34] de RackSpace et en collaboration avec la NASA.

Le marché du Cloud n'est encore qu'à ses débuts et d'après une étude menée par Forrester [35], le marché du Cloud Computing s'élevait à environ 5.5 milliards de

[4] Est un service de plateforme en tant que service de la société RedHat.
[5] Est un ensemble de logiciels open source permettant de déployer des infrastructures de cloud computing.

par Bernstein et Celesti [37]. Des recherches récentes ont porté sur l'aide d'un environnement Cloud [38] qui contrôle plusieurs Clouds en évitant leurs dépendances.

Le rapport du NIST [39] a déclaré que le multi-Cloud peut être utilisé en série lorsqu'une application ou service se déplace d'un Cloud vers un autre ou lorsque les services hébergés sur différents Clouds sont utilisés simultanément.

Les scénarios les plus simples sont les migrations des applications d'un Cloud privé vers un Cloud public (pour l'utilisation en série) ou des applications qui utilisent plusieurs services hébergés dans différents Clouds publics (pour l'utilisation simultanée). Il existe plusieurs raisons pour lesquelles les ressources et services provenant de plusieurs Clouds sont devenu autant nécessaires et diverses [40], ces raisons sont mentionnées dans divers rapports de recherche et documents industriels [41, 42, 43]. L'auteur du papier [40] a fait un résumé non exhaustif en se concentrant essentiellement sur celles qui apparaissent à plusieurs reprises dans différents cas d'utilisation. Les différentes raisons recensées par l'auteur [40] sont reportées dans le (tableau 2), ce tableau mentionne les deux cas d'utilisation de multi Cloud (série et simultané) qui sont utilisés dans le rapport NIST. L'adoption des multi-Cloud dépend de l'intérêt ou le besoin des acteurs. Les principaux acteurs dans les scénarios de cas d'utilisation sont : le fournisseur de Cloud, l'utilisateur ou le consommateur de Cloud, le développeur d'applications et le courtier de Cloud [40].

Type d'utilisation	**Raison**
Utilisation en série	Optimiser les coûts ou améliorer la qualité des services
	Réagir à l'évolution de l'offre des fournisseurs
	Prendre en compte des contraintes comme l'emplacement
	Éviter la dépendance à un seul fournisseur externe
	Assurer la sauvegarde pour faire face aux catastrophes ou la planification de l'inactivité
Utilisation simultanée	Agir comme un intermédiaire

	Faire face aux pics de services et à la demande de ressources en utilisant des services externes à la demande
	Réplication d'application ou service en utilisant différents Clouds pour garantir leur disponibilité
	Amélioration de l'infrastructure Cloud en passant des accords avec d'autres fournisseurs
	Consommation de différents services en fonction de leur particularité qu'on ne trouve pas ailleurs

Tableau 2 : Le top dix des raisons pour utiliser les multi-Cloud [40]

I.7.3. Inter-Cloud :

L'Inter-Cloud est un réseau global et une extension d'internet "network of networks" [44]. Inter-Cloud Computing est l'interconnexion des infrastructures de plusieurs fournisseurs de Cloud.

L'accent est mis sur l'interopérabilité entre les fournisseurs de services de Cloud public, pour fournir des services Cloud avec succès, les interconnexions entre Cloud sont nécessaires, et l'interopérabilité ainsi que la portabilité sont des facteurs importants de l'inter-Cloud [45].

Une limite majeure du Cloud est que les systèmes Clouds ont peu de ressources physiques. Si un Cloud a épuisé toutes les ressources de calcul et de stockage, il ne peut pas fournir les services à la clientèle. L'Inter-Cloud traite des situations où chaque Cloud utilise le stockage informatique, ou tout type de ressources des infrastructures d'autres Clouds [46]. L'environnement d'inter-Cloud offre plusieurs avantages comme la diversification des lieux géographiques, aussi, il permet d'éviter l'enfermement propriétaire vers le Cloud client. Les avantages pour le fournisseur de services Cloud sont l'expansion à la demande et des accords de niveau de service (SLA) pour les clients [47].

I.8. Conclusion :

Le Cloud Computing est actuellement un concept en plein essor. Nous avons passé en revue, dans ce chapitre les notions de base relatives à ce concept qui sont les caractéristiques, les modèles de déploiement ainsi que les trois modèles de service du Cloud Computing, à savoir SaaS, PaaS et Iaas, ensuite nous avons listé un ensemble d'avantages qu'offre ce dernier.

Malgré ses avantages, beaucoup de problèmes restent en suspens et suscitent l'intérêt des chercheurs en informatique.

Par la suite, nous avons parlé sur l'évolution du Cloud Computing qui dépend de l'évolution du multi-Cloud et de l'inter-Cloud.

Dans le chapitre suivant, nous allons présenter et discuter le problème d'interopérabilité entre les systèmes d'informations.

Chapitre II. *Interopérabilité entre SI*

II.1. Introduction :

Les systèmes d'information (SI) sont une brique essentielle à l'organisation des entreprises. Ils sont aujourd'hui construits à partir de l'agrégation de systèmes informatiques qu'il convient de maintenir et faire évoluer avec agilité et sans entropie. Les objectifs sont alors entre autres d'améliorer la qualité des services offerts tout en préservant l'autonomie des acteurs, l'ouverture des SI, une gestion cohérente des informations, des temps de production réduits et une meilleure maitrise des couts de maintenance. Cependant, face à la complexité des SI, la mise en place de l'interopérabilité entre SI est difficile à la fois aux niveaux conceptuels et techniques.

Ce chapitre présente un aperçu sur les notions de bases de l'interopérabilité entre SI et ses différentes facettes.

II.2. Formes de collaboration entre SI :

Bien que chacun ait une notion intuitive de ce que la collaboration signifie, ce concept est souvent confondu avec celui de la coopération. Souvent, ces deux termes sont utilisés pour désigner la même chose. L'ambiguïté atteint un niveau plus élevé lorsque l'on considère d'autres termes proches comme la mise en réseau (Networking), la communication et la coordination [48,49].

Bien que chacun de ces concepts soit un prérequis important pour la collaboration, ils ne lui sont pas équivalents. Pour clarifier ces diverses notions, nous reprenons les définitions suivantes issues de Camarinha-Matos et Afsarmanesh [50] qui sont synthétisées dans la (figure 3) :

II.2.1. Networking :

Implique la communication et l'échange d'informations pour réaliser un avantage commun.

II.2.2. Coordination :

En plus du networking, elle implique l'alignement et la synchronisation des activités mises en commun pour rendre plus efficaces les résultats attendus.

II.2.3. Coopération :

Elle implique non seulement la coordination, mais aussi le partage des ressources pour réaliser des objectifs compatibles. La coopération est réalisée à travers la répartition des tâches à réaliser entre les participants.

II.2.4. Collaboration :

C'est un processus dans lequel les entités partagent des informations, des ressources et des responsabilités pour planifier conjointement, mettre en œuvre et évaluer un programme d'activités leur permettant de réaliser des missions communes. La collaboration implique l'engagement mutuel des participants pour résoudre ensemble un problème. Ceci implique une confiance mutuelle.

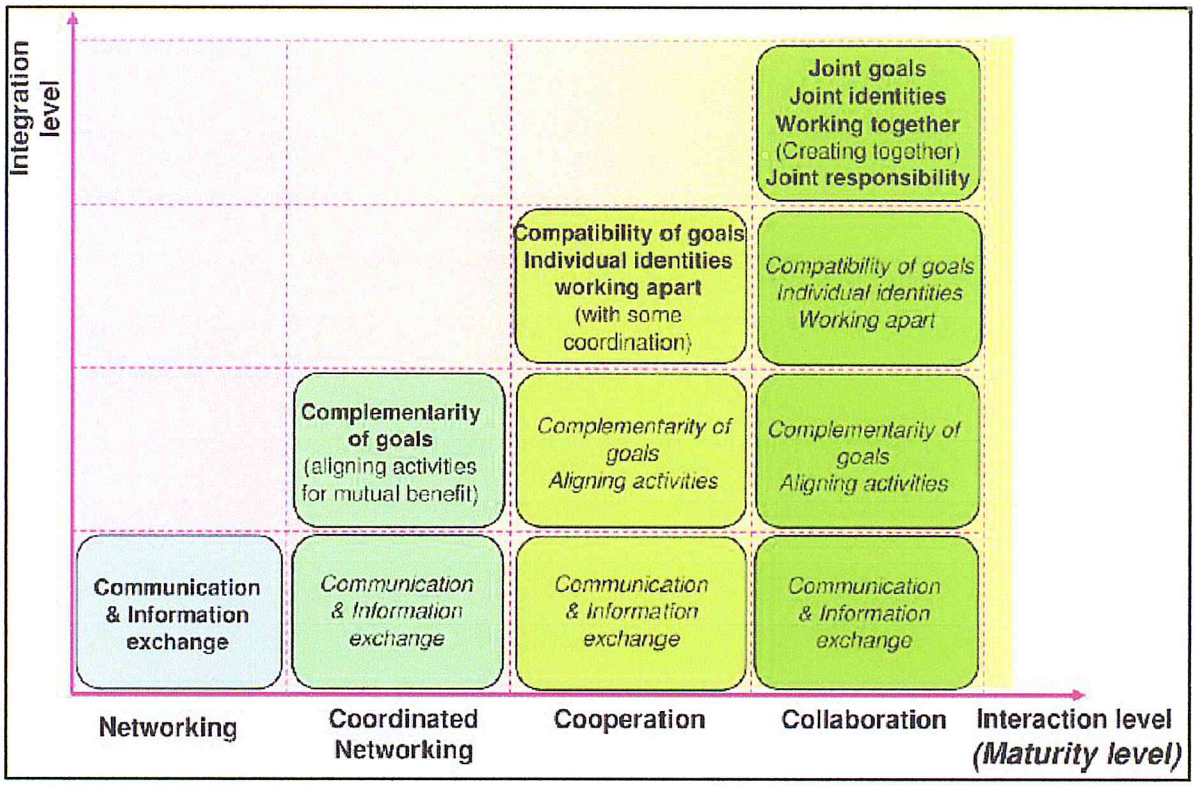


Figure 3 : Stratégies mises en place par les entreprises en réseau [50]

La conception d'une collaboration joue un rôle majeur pour le développement des entreprises car elle permet selon Cassier [51] de travailler à une œuvre commune avec un but général partagé, mais la réussite des collaborations dépend aussi bien de la politique de pilotage de ces collaborations, de la qualité de communication et des capacités d'interaction mises en œuvre par chaque partenaire. Du point de vue de pilotage, les concepts d'organisation virtuelle (OV) et d'entreprise étendue offrent de nouveaux modèles de coopération entre les différentes entreprises en leur permettant de gérer les projets de développement de produits à travers toute la chaîne logistique. Comme toute organisation, la définition et le pilotage d'une OV doit couvrir tous les niveaux stratégiques, tactiques et opérationnels [52].

Le caractère dynamique et évolutif des collaborations implique à chaque partenaire (donneur d'ordre ou fournisseur) de pouvoir contrôler ses dépendances vis-à-vis des autres partenaires. Ceci permet aussi d'éviter des profonds changements internes en termes d'outils et de méthodes de travail tout en prévoyant une flexibilité des processus et des capacités de communications des systèmes d'information pour s'adapter à tout type de situation[52].

Face au caractère complexe et contraignant des problématiques de collaboration, le concept d'interopérabilité s'impose de plus en plus comme une solution pertinente pour favoriser les échanges d'informations entre des systèmes hétérogènes. En effet, l'intérêt majeur de l'interopérabilité réside dans la capacité à développer des interfaces de communication entre des organisations tout en leur garantissant un fonctionnement indépendant les uns par rapport aux autres [52].

II.3. Définitions de l'interopérabilité :

Depuis novembre 2000, sur l'initiative de l'OMG, l'ingénierie dirigée par les modèles a proposé différentes approches et technologies pour le développement et la maintenance des systèmes a prépondérance logicielle. Dans ce contexte, un des objectifs visés était de pallier les difficultés d'interopérabilité entre les systèmes en s'appuyant sur les modèles au lieu des intergiciels. Depuis, les architectures à base de services sont venues compléter le tableau en mettant en avant des standards de développement et de communication visant à favoriser les intégrations des systèmes et une meilleure agilité par l'adaptation des systèmes.

La littérature offre plusieurs définitions différentes de l'interopérabilité Baïna [53] dresse un inventaire de ces définitions. Selon les auteurs, l'interopérabilité est :

1. La capacité de deux ou plusieurs systèmes ou composants à échanger des informations et à utiliser les informations échangées [54] ;
2. Réussie si et seulement si l'interaction entre les systèmes impliqués couvre les aspects donnés, ressources et processus métiers en utilisant les sémantiques définies dans le domaine métier [55] ;
3. La capacité à communiquer avec des systèmes pairs et accéder à leurs fonctionnalités [56] ;
4. L'aptitude de deux systèmes (ou plus) à communiquer, coopérer et échanger des données et services, malgré les différences dans les langages, les implémentations, les environnements d'exécution ou les modèles d'abstraction [57] ;
5. Dans le domaine des sciences de l'information, elle correspond à la capacité des systèmes à échanger et utiliser des informations (généralement dans un réseau hétérogène composé de plusieurs réseaux locaux).

Il existe diverses définitions de l'interopérabilité. Dans ce mémoire, nous considérons, dans un premier temps, la définition d'IEEE [58] qui définit l'interopérabilité comme *la capacité de deux systèmes ou plus à échanger de l'information et à l'utiliser.* Carney, Fisher et Place [59] étendent cette définition en ajoutant la notion d'objectif lié à l'interopération ainsi que la notion de contexte qui définit l'environnement dans lequel évolue l'ensemble des informations échangées. Finalement, nous avons fini par adopter la définition de [59] qui définissent l'interopérabilité comme *la capacité d'un ensemble d'entités communicantes à échanger de l'information spécifique et à opérer à partir de cette information selon une sémantique commune, dans le but d'accomplir une mission spécifiée dans un contexte donné.*

II.4. Considérations de l'interopérabilité :

Pour pouvoir atteindre l'objectif d'interopérabilité plusieurs aspects doivent être pris en considération. Dans ce qui suit nous allons résumer les différentes considérations de l'interopérabilité :

II.4.1. Préoccupations de l'interopérabilité :

Dans l'étude du concept de collaboration, il est indiqués que dans les collaborations, les acteurs échangent ou partagent différentes entités telles que les données ou les informations à l'aide de différents outils de communication. Par exemple, un échange ou partage peut inclure un transfert manuel d'un document d'un acteur à un autre ou il pourrait impliquer le partage d'un service via une application. Selon Chen [60], les préoccupations de l'interopérabilité, considère les entités échangées entre les acteurs. Les classifications des préoccupations sont fondées sur les contenus des entités échangées ou sur le niveau d'interaction. Quatre niveaux sont identifiés par Chen : données, services, processus et business : « in an enterprise, data is used by services (or functions to provide a service). Services (functions/activities) are employed by processes to realize business of the enterprise » [61].

En parallèle à la classification proposée par Chen et Obrst [62] classifie les entités échangées en six niveaux : donnée/information, composant, application, service, entreprise et communauté. Nous allons prendre en considération la classification proposée par Interoperability Working Group [63], qui classe les

préoccupations de l'interopérabilité en quatre catégories : procédure, application, infrastructure et donnée :

II.4.1.1. Procédure :

Ce niveau comprend les pratiques, les architectures et les normes pour l'échange d'informations.

II.4.1.2. Application :

Ce niveau comprend les logiciels pour l'échange, le traitement et la manipulation de l'information.

II.4.1.3. Infrastructure :

Ce niveau représente l'environnement technique (matériel, réseau, système) qui permet les interactions entre les applications.

II.4.1.4. Donnée :

Ce niveau représente les formats et les protocoles d'échanges d'informations à travers une sémantique commune.

II.4.2. Niveaux d'interopérabilité :

Nous considérons les niveaux d'interopérabilité cités dans EIF [64]. L'interopérabilité peut se produire à plusieurs niveaux :

II.4.2.1. Niveau technique (données et messages échangés) :

Pour pouvoir échanger des informations, il faut au préalable s'assurer que le transport des données d'un système à un autre soit possible, c'est-à-dire qu'il existe un vecteur des données fonctionnel. L'exemple de deux personnes en communication téléphonique permet d'illustrer ce point de vue : pour que des paroles soient échangées, il est nécessaire que les deux personnes puissent parler, entendre, et que leurs téléphones/réseau soient en état de fonctionnement. C'est ce que sous-tend l'expression « capacité à échanger ». Le niveau technique est une condition nécessaire mais non suffisante pour établir l'interopérabilité.

II.4.2.2. Niveau sémantique (information et partage de services) :

Un vecteur d'information fonctionnel ne suffit pas, encore faut-il que les données échangées soient comprises par les deux systèmes. Les données, chargées de sens deviennent alors des informations qui peuvent être traitées par les systèmes en question. Pour l'exemple de personnes en communication téléphonique, les deux parties doivent être capables d'interpréter le sens du message de leur interlocuteur (nous retrouvons la notion de système interprétatif de Tsuchiya [65]). C'est ce que sous-tendent les expressions «utiliser l'information» ou «niveau sémantique».

II.4.2.3. Niveau organisationnel (interactions business unit/processus/personnes à travers de l'organisation) :

Une organisation adaptée doit être prévue pour assurer l'échange des informations. Dans le cas de la communication téléphonique, l'échange des informations ne sera pas possible si l'une des personnes est absente. Il sera perturbé si l'une d'elles est occupée simultanément par une autre activité. C'est ce que sous-tendent les expressions « interopérabilité structurale » ou « niveau organisationnel ».

Il faut veiller à satisfaire simultanément tous ces niveaux pour garantir une interopérabilité complète entre SI.

II.4.3. Barrières à l'interopérabilité :

La collaboration est un processus qui assure l'interopération entre les systèmes, et pour assurer cette dernière il faut résoudre les problèmes d'interopérabilité qui sont des barrières qui s'opposent à sa réalisation. Ces barrières à l'interopérabilité ont été identifiées par Daclin et Chapurlat [66] comme suit :

II.4.3.1. Les problèmes d'ordre technologique :

Ils sont relatifs à l'incompatibilité des technologies de l'information (plateformes et applications informatiques). Ces problèmes concernent les normes pour présenter, stocker, échanger, traiter et communiquer les données via les moyens informatiques.

II.4.3.2. Les problèmes de type conceptuel :

Concernent la syntaxe et la sémantique des informations qui sont échangées. Ces problèmes concernent la modélisation des informations à un haut niveau d'abstraction comme par exemple un modèle d'entreprise, ainsi que la façon de structurer les informations en vue d'un échange.

II.4.3.3. Les problèmes d'ordre organisationnel :

Ils sont liés à la définition de la responsabilité et d'autorité qui induisent ou impactent la qualité des conditions de travail. Ce sont donc essentiellement les facteurs humains et les comportements organisationnels qui peuvent être incompatibles avec l'interopérabilité.

La résolution des barrières d'ordre technologique est désormais simplifiée et partiellement atteinte par les interfaces techniques et les standards mis en place. Nous pouvons citer, par exemple, XML (eXtensible Mark-up Language) et les applications qui lui sont attachées : SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), ebXML (Electronic Business XML Initiative) ainsi que les diverses initiatives de standardisation [67, 68,69].

Les barrières organisationnelles restent de nos jours une difficulté majeure qui pose le problème de l'accès au niveau de l'interopérabilité désiré.

L'interopérabilité sémantique, dans la mesure où elle traite de la signification des informations échangées, représente une piste de recherche intéressante qui reste très ouverte aujourd'hui. Dans ce mémoire, nous nous focalisons donc principalement sur l'étude de l'interopérabilité sémantique.

Dans ce contexte, plusieurs initiatives ont été menées par diverses organisations pour contribuer à résoudre la problématique d'interopérabilité sémantique. Nous pouvons citer notamment :

- **US Government:** Healthcare Information Technology Enterprise Integration. Le but est d'augmenter l'efficience et la productivité des SI dans le domaine de la santé [70].
- **Commission Européenne:** initiatives eEurope 2005 et i2010. Le but est d'assurer l'interopérabilité des e-services gouvernementaux : IDABC11

(Interoperable Delivery of pan-European Government Services to public Administrations, Business and Citizens).

II.4.4. Approches de l'interopérabilité :

La norme ISO-14258-1998 (1998) [71], relative à la modélisation des entreprises, précise que l'interopérabilité entre deux (ou plusieurs) systèmes d'information d'entreprise peut être abordée de trois manières :

- **Intégration :**

Un standard commun de modèle de données est utilisé pour tous les composants du système. Le processus d'intégration revient à fusionner les modèles de données ;

- **Unification :**

Un méta-modèle commun à tous les composants du système fournit un moyen pour établir des correspondances sémantiques ;

- **Fédération :**

Des modèles distincts sont associés dynamiquement. Cette approche s'appuie habituellement sur des outils semi-automatiques, basés sur des méthodes heuristiques qui comparent principalement la terminologie et la structure des données afin de détecter les couples de concepts qui sont reliés au niveau sémantique (similarité ou équivalence).

Hoffmann [72] à son tour définit les approches de l'interopérabilité de la manière suivante :

II.4.4.1. L'approche d'intégration :

Requiert de développer une nouvelle ontologie qui reflète un consensus sur le point de vue des différentes organisations qui collaborent, afin d'en limiter les incohérences. En raison de ces compromis, la nouvelle ontologie pourra n'avoir qu'une compatibilité limitée avec les ontologies source des différentes organisations. D'autre part, ce qui n'est pas signalé par Hoffmann [72], la fusion des modèles de données suppose de les connaître complètement, ce qui n'est souvent pas le cas lorsque

le problème d'interopérabilité implique des applications propriétaires dont les modèles de données sont considérés comme des secrets industriels.

II.4.4.2. L'approche d'unification :

La plupart des collaborations ne seront pas connues avant que les ontologies ne soient développées, cette approche requiert de modifier les ontologies en accord avec l'ontologie de plus haut niveau. Comme il peut y avoir plusieurs ontologies de plus haut niveau qui soient pertinentes, cette approche amènera à développer plusieurs versions d'ontologie, qu'il faudra les garder à jour.

II.4.4.3. L'approche de fédération :

Permet l'échange parmi des ressources développées dans des buts indépendants et qui évoluent de manière indépendante; elle semble donc la plus approprié. Cependant malgré tous les efforts investis dans les méthodes et outils pour l'alignement d'ontologies, les résultats sont encore décevants.

Le choix de l'une ou l'autre de ces approches est critique dans le cas de conservation du flux sémantique dans un contexte de mise à l'échelle. La (figure 4) illustre les trois approches principales de l'interopérabilité :

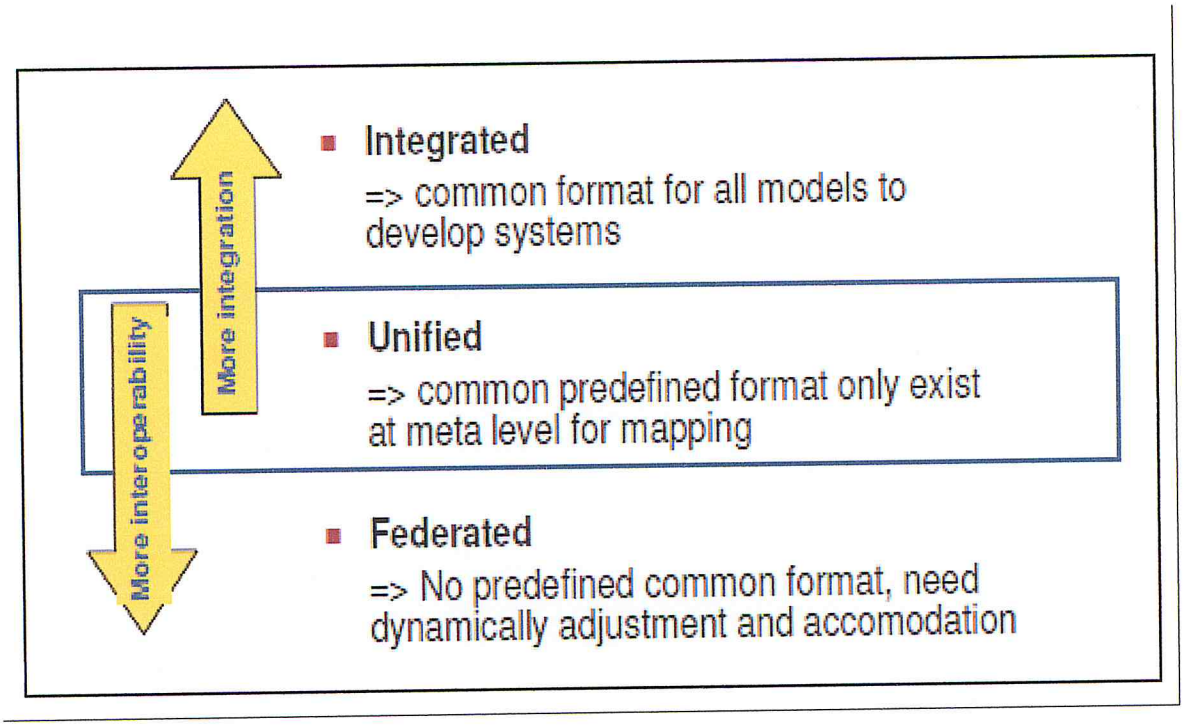


Figure 4 : Approches de base pour développer l'interopérabilité [73]

Les trois approches développent l'interopérabilité entre les systèmes d'entreprises. L'état fédéré est considéré comme le plus important des approches car il permet de développer une interopérabilité totale. Toutefois, le choix dépend du contexte et des besoins. Si la nécessité d'interopérabilité vient d'une fusion d'entreprises, l'approche intégrée semble être la plus adaptée, dans ce cas, il n'y a qu'un format commun pour tous les partenaires, et tous les modèles sont construits et interprétés en fonction de celui-ci. Si la nécessité d'interopérabilité concerne une collaboration à long terme, l'approche unifiée semble une solution possible, pour cela, un méta-modèle commun à travers les modèles des partenaires fournit un moyen d'établir l'équivalence sémantique. Enfin, pour un besoin d'interopérabilité issue d'un projet de collaboration à court terme (ex. entreprise virtuelle) ; l'approche fédérée peut être utilisée. Pour inter-opérer les partenaires doivent s'adapter dynamiquement pour parvenir à un accord [74].

La (figure 5) résume les concepts d'interopérabilité entre les systèmes d'entreprises dans le but de définir une ontologie de l'interopérabilité d'entreprise [75].

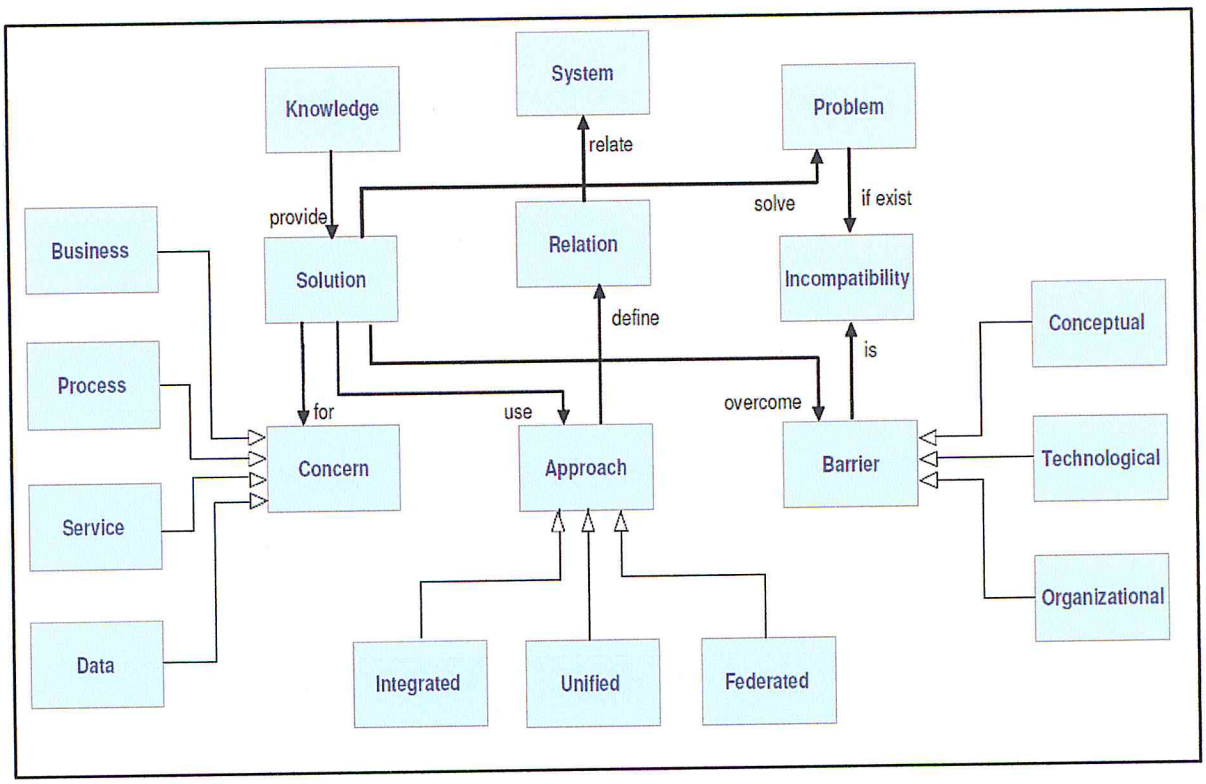


Figure 5 : Vue d'une ontologie de l'interopérabilité des entreprises [75]

II.4.5. Espace de problème et espace de solution :

Selon Chen [60], l'espace de problème d'interopérabilité est constitué des deux dimensions : préoccupations et barrières de l'interopérabilité voir la (figure 6). L'intersection des deux dimensions est l'ensemble des problèmes d'interopérabilité ayant les mêmes obstacles et la même préoccupation. Les trois dimensions de l'interopérabilité : préoccupations, obstacles et approches constituent ensemble la solution d'interopérabilité.

La (figure 6) résume l'espace problème et l'espace solution de l'interopérabilité selon les différentes dimensions.

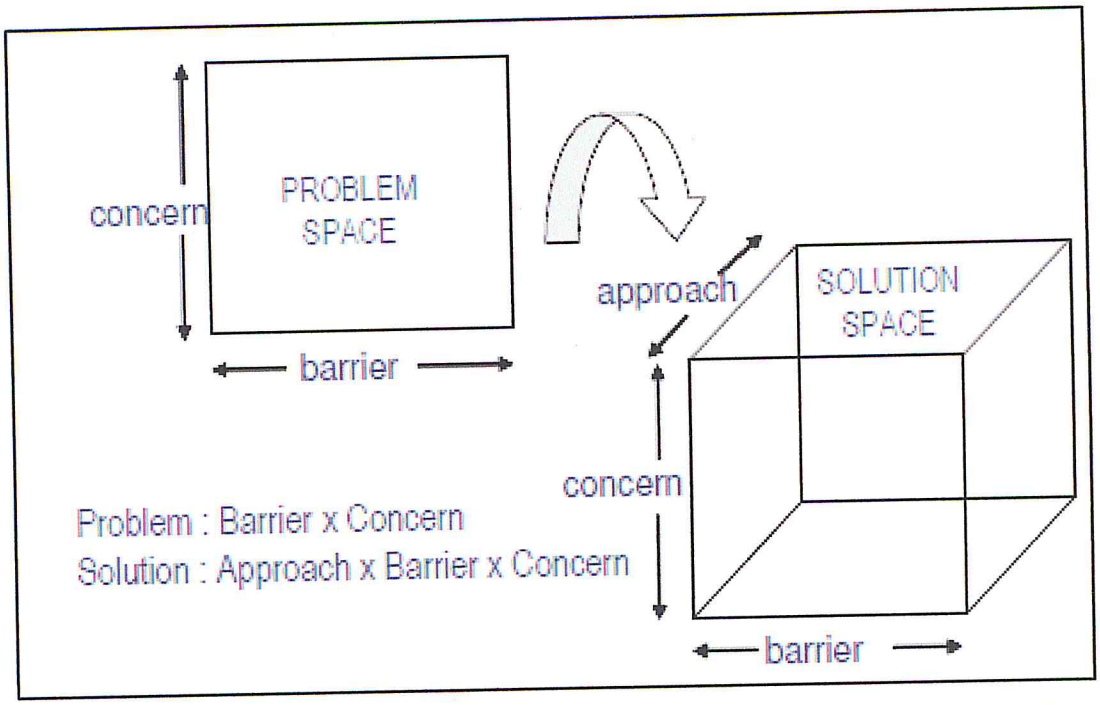


Figure 6 : Espace de problème et espace de solution de l'interopérabilité [60]

II.5. Conclusion :

L'interopérabilité des systèmes d'information est de plus en plus exigée à cause du besoin continu d'intégration de nouveaux systèmes et des systèmes en évolution, en particulier dans le contexte des réseaux d'entreprises collaboratives.

Un de ses enjeux est de faire collaborer et/ou interagir des applications distribuées, autonomes et hétérogènes. Cette problématique a fait l'objet de nombreux travaux scientifiques et de recherches visant à fournir des solutions génériques.

Nous avons présenté, dans ce chapitre, des notions de base de l'interopérabilité qui sont : les types, les préoccupations, les besoins, les barrières et les approches d'interopérabilité. Dans le chapitre suivant nous allons explorer l'interopérabilité entre Clouds.

Chapitre III. *Interopérabilité dans le Cloud Computing*

III.1. Introduction :

Ce chapitre présente des informations de base et une revue de littérature sur les travaux connexes de l'interopérabilité dans le Cloud Computing. Il est essentiel de recueillir des informations sur les recherches passées et de comprendre les problèmes et les défis actuels dans le domaine de l'interopérabilité sémantique du Cloud Computing avant de pouvoir proposer une solution traitant ce domaine. Dans ce chapitre, nous présentons tout d'abord un aperçu sur l'évolution de l'interopérabilité pour la technologie du Cloud Computing, ensuite nous explorons les deux dimensions de l'interopérabilité ainsi que les normes et les modèles d'interopérabilité existants. En dernier, nous discuterons les expériences existantes réalisant l'interopérabilité sémantique dans l'environnement du Cloud Computing.

III.2. Interopérabilité entre Clouds :

Les solutions de Cloud Computing existantes n'ont pas été construites en tenant compte de l'interopérabilité [76]. Elles verrouillent généralement les clients dans une infrastructure, une plate-forme ou un service Cloud unique empêchant la portabilité des données ou des logiciels créés [77]. En outre, la bataille pour la domination entre les grands fournisseurs, comme Amazon, Google et SalesForce, les rend réticents à se mettre d'accord sur des normes largement acceptées pour promouvoir leurs propres formats incompatibles [78]. Cette domination augmente l'effet de verrouillage et affecte la concurrence puisque les petites et moyennes entreprises (PME) s'opposent à entrer dans le marché Cloud. L'Agence européenne pour la sécurité des réseaux et de l'information (ENISA) et la Commission européenne

ont reconnu le problème de verrouillage des fournisseurs comme un risque élevé que comportent les infrastructures Cloud [79].

L'interopérabilité est l'élément manquant qui remédiera à cette situation et bénéficiera à la fois aux clients et aux fournisseurs Cloud. En particulier, dans un environnement Cloud interopérable, les clients pourront comparer et choisir des offres parmi les différents offres Cloud de différentes caractéristiques, en négociant avec des fournisseurs Cloud à chaque fois que cela s'avérerait nécessaire, sans mettre en danger les données et les applications. De plus, un marché interchangeable de Cloud ouvrira l'industrie informatique aux PME et renforcera leurs positions sur le marché. Elles inter-opéreront et coopéreront en créant de nouveaux modèles commerciaux selon la demande sans conflits en raison de problèmes d'interopérabilité [80].

III.3. Dimensions conceptuelles de l'interopérabilité :

L'interopérabilité dans le Cloud Computing peut être conceptualisée selon deux dimensions : verticale et horizontale.

III.3.1. Dimension verticale :

La dimension verticale décrit comment le Cloud Computing facilite l'interopérabilité au sein d'une seule plateforme, par exemple entre différents appareils et applications utilisés par le même consommateur ou utilisateur final. Lorsque quelqu'un utilise Gmail, par exemple, il est possible d'accéder à ses messages électroniques depuis n'importe quel ordinateur connecté à internet, y compris les smartphones.

La dimension verticale est importante car elle facilite l'indépendance de l'appareil et l'emplacement, deux éléments qui sont intensivement liés. Lorsqu'un utilisateur accède à ses données ou ses services à partir du Cloud, il n'est plus un obstacle à l'ordinateur unique qui autrement stockait les données ou exécutait l'application, en général, tout périphérique avec une connexion internet suffit largement. Puisque le traitement et le stockage peuvent être effectués sur le Cloud, l'utilisateur peut utiliser des programmes ou accéder aux informations des clients légers, relativement peu performants (par exemple, des ordinateurs ou des programmes

qui s'appuient sur des serveurs connectés au réseau pour la plupart de leurs besoins de traitement), tels que les netbooks et les smartphones [81].

De même, les deux facteurs contribuent à l'indépendance de la localisation : la possibilité d'accéder à des données et des applications avec n'importe quel périphérique connecté à Internet, quel que soit son emplacement. De telles fonctionnalités de traitement et de stockage à distance peuvent s'avérer particulièrement utiles dans des pays en développement, où l'accès à l'infrastructure technologique peut être limité [82].

III.3.2. Dimension horizontale :

La dimension horizontale concerne l'interopérabilité entre les services concurrents de Cloud Computing. Elle décrit à quel point une entreprise possédant son logiciel hébergé dans un service Cloud peut se déplacer vers un fournisseur de Cloud concurrent qui offre des tarifs plus favorables ou un service plus fiable. Elle exige que les moyens cohérents de coordination entre les produits Cloud soient disponibles, tels que les contrats conventionnels, les arrangements, les fonctionnalités de sécurité, la confidentialité des données ou les moyens de gestion de l'identité.

La dimension horizontale peut concerner à la fois les consommateurs et les entreprises qui achètent des produits Cloud [81]. Elle inclut non seulement des barrières techniques pour passer les produit Cloud d'un service à un autre, mais aussi des incompatibilités ou des incohérences entre la confidentialité des données et les politiques de sécurité, la gestion de l'identité pour les différents fournisseurs de Cloud [81]. Par exemple, pour un hôpital qui impose des lois de confidentialité strictes, il faudrait que ses fournisseurs de services Cloud mettent en place de solides règles de confidentialité et des normes de sécurité [83].

III.4. Standards de l'interopérabilité :

À ce jour, la majeure partie de l'attention portée aux normes d'interopérabilité des Clouds a été appliquée à la couche IaaS, bien que l'activité au niveau PaaS commence à s'accélérer. En outre, il existe plusieurs normes de sécurité qui permettent et facilitent l'interopérabilité dans le Cloud Computing. Dans ce qui suit nous présentons quelques standards de l'interopérabilité dans le Cloud Computing :

III.4.1. Open-SCA :

Le modèle d'assemblage CSA est une solution pour décrire la coordination et l'interaction des services qui se lient dans la composition du service avec des préoccupations d'architecture logicielle commune. Les normes CSA peuvent être utilisées tel quel ou peuvent servir d'entrée pour toute composition et langage d'assemblage. Les spécifications du modèle d'assemblage CSA sont très pertinentes pour l'interopérabilité. L'interopérabilité des données est une préoccupation tout aussi importante dans Open-CSA [84].

III.4.2. USDL :

La définition de l'USDL vise à compléter la pile de langage technique en ajoutant les informations commerciales et opérationnelles requises. Les acteurs du Cloud ciblés pour USDL sont des fournisseurs de services, des fournisseurs d'infrastructure, des assembleurs de services et des consommateurs de services. USDL définit un langage centré sur l'interopérabilité qui permet à ses utilisateurs de modéliser des services arbitraires et de s'intégrer aux normes existantes. L'objectif d'aborder la modélisation des services et les mappages de support aux différentes normes rend cela intéressant pour la modélisation interopérable des Clouds. Cela permet de nouveaux modèles commerciaux pour le courtage de services car les services peuvent être proposés, livrés, exécutés et composés automatiquement à partir de services de différents fournisseurs [85].

III.4.3. EMML (Mashups) :

EMML est conçu pour être complémentaire et intégré à des langages tels que JavaScript, Java, Groovy et Ruby via des scripts. Il est particulièrement adapté aux problèmes d'interopérabilité liés à la création de Mashup. Il prend en charge un montage léger et intégratif des services et, par conséquent, représente des préoccupations spécifiques en matière de modélisation et d'intégration [86].

III.4.4. OCCI :

Le groupe de travail OGF OCCI fournit une spécification API pour la gestion de l'infrastructure de Cloud Computing pour prendre en compte ces préoccupations

[87]. La portée est une gamme complète de fonctionnalités de haut niveau requises pour la gestion du cycle de vie des machines virtuelles (ou des charges de travail) fonctionnant sur les technologies de virtualisation (comme les conteneurs) supportant l'élasticité du service. OCCI fournit une API pour l'interfaçage des installations IaaS de Cloud Computing, ce qui est suffisamment complet pour faciliter la mise en œuvre des implémentations interopérables. Tout en ciblant les préoccupations d'IaaS, il peut favoriser les efforts d'interopérabilité à des niveaux supérieurs. La portée d'OCCI est une fonctionnalité de haut niveau requise pour la gestion du cycle de vie. Ceci est en partie réalisé grâce à la couverture des API propriétaires existantes [87].

III.4.5. CIMI :

CIMI qui traite de la maintenance et de l'approvisionnement en temps réel des services Cloud. La portée de la norme CIMI couvre la fonctionnalité IaaS principale, le déploiement et la gestion des machines virtuelles et d'autres artefacts tels que les volumes, les réseaux ou le suivi. Une fois interfacé avec le fournisseur IaaS, les informations qui doivent être traitées pour gérer un service Cloud peuvent être découvertes de façon itérative, y compris les métadonnées décrivant les capacités et les contraintes de ressources. La plupart des développeurs utilisent le protocole CIMI REST / HTTP, l'interface actuelle reliant le modèle (d'autres sont attendus plus tard) [88]. Cela fournit des codes d'état HTTP standard et prend en charge les formats de sérialisation JSON et XML [88].

III.4.6. Cloud Application Management for Platforms (CAMP) :

En tant que premier effort pour standardiser une interface de gestion PaaS, CAMP est destiné à fournir une base commune pour le développement d'outils de gestion multi-Cloud ainsi que pour offrir aux fournisseurs de Cloud et au clients une approche basée sur les REST pour la gestion des applications. CAMP fait progresser un protocole interopérable que les développeurs de Cloud peuvent utiliser pour emballer et déployer leurs applications. Il fournit un vocabulaire de développement commun et une API qui peut fonctionner à travers de multiples Cloud sans adaptation excessive [89].

III.4.7. TOSCA :

Soutenu par OASIS, le cadre de TOSCA vise à améliorer la portabilité des applications et des services Cloud. TOSCA permet une description interopérable des services de l'application et de l'infrastructure Cloud, des relations entre les parties du service et le comportement opérationnel de ces services (déploiement, patch, arrêt) indépendants du fournisseur qui crée le service et de tout fournisseur de Cloud particulier ou technologie d'hébergement [88].

III.4.8. CDMI :

CDMI est une norme pour l'auto-provisionnement, l'administration et l'accès au stockage d'un Cloud. CDMI définit les opérations RESTful HTTP pour accéder aux fonctionnalités du système de stockage Cloud. CDMI définit l'interface fonctionnelle que les applications utilisent pour créer, récupérer, mettre à jour et supprimer des éléments de données du Cloud. Dans le cadre de cette interface, un client peut découvrir les capacités de l'offre du stockage Cloud et utiliser cette interface pour gérer les conteneurs et les données qui y sont placées. En outre, les métadonnées peuvent être définies sur les conteneurs et leurs éléments de données contenus via cette interface [88].

III.4.9. OVF :

Le format Open Virtualisation (OVF) décrit un format ouvert, sécurisé, portable, efficace et flexible pour la distribution d'une ou plusieurs machines virtuelles [90]. Les fonctionnalités OVF comprennent une distribution optimisée et la portabilité des appareils virtuels. Il prend en charge la compression pour des transferts de paquets plus efficaces, la vérification du contenu et la vérification de l'intégrité, et fournit un schéma de base pour la gestion des licences de logiciels. Il prend en charge l'indépendance des fournisseurs et des plateformes, car il ne dépend pas de l'utilisation d'une plate-forme hôte spécifique, d'une plate-forme de virtualisation ou d'un système d'exploitation hôte ou invité. OVF est un format portable qui permet aux utilisateurs de déployer des machines virtuelles dans n'importe quel hyperviseur qui supporte OVF.

En plus des normes, il existe un certain nombre de projets open source comme OpenStack, Cloud Foundry et OpenShift qui ont un impact positif sur l'interopérabilité du Cloud Computing.

III.5. Modèles d'interopérabilité sémantique entre Clouds :

III.5.1. Modèle d'ontologie (SOA) :

Dans les Cloud interopérables sémantiquement RASIC, les modèles et technologies sémantiques seront employés pour deux raisons principales. D'une part, ils résolvent les conflits d'interopérabilité sémantique qui sont augmentés lorsque différentes plates-formes Cloud échangent des données. D'autre part la sémantique sera utilisée à la couche SOA afin de fournir des moyens pour développer la découverte et l'interopérabilité des services intelligents et des mécanismes de recommandation. La couche sémantique comprend le service léger et modes de ressources informatiques, service et ressource composants, interopérabilité sémantique Run-time Engine. Le modèle de service constitue un système simple et ouvert et un vocabulaire extensible pour décrire les services SOA. Les modèles de service et de ressources ne seront pas développés à partir de zéro et la réutilisation des efforts existants sera envisagée, par ex [91].

Les services et les modèles de ressources seront officiellement exprimés en ontologies en utilisant un langage d'ontologie standardisé [91].

III.5.2. Cloud Pattern :

Cloud Patterns peut être considéré comme une autre catégorie de patron, en se concentrant sur la description des problèmes et des solutions liés au Cloud Computing. Leur développement est toujours en une étape précoce, aucun formalisme ou outil n'a été encore adopté [93]. Néanmoins, différents catalogues en ligne ont été publiés, à la fois par des fournisseurs de Cloud [94] [95] ou en conséquence de la recherche académique [96] [97].

Les propriétaires des patterns fournissent des indications sur les services Cloud ou les composants particuliers qui peuvent être utilisés pour mettre en œuvre des fonctionnalités, se référant évidemment à la plate-forme à laquelle ils ont été conçus.

Plusieurs formalismes ont été proposés pour décrire les modèles, mais aucun d'entre eux n'a été largement adopté. Nous allons présenter quelques formalismes pour la représentation de pattern [93].

III.5.2.1. Modèle d'automate :

La base de connaissances basée sur la représentation sémantique, peut être facilement interrogée par les moyens de SPARQL [98], un langage de requête pour OWL (n'est pas un langage d'inférence comme SWRL). L'utilisation des requêtes SPARQL simplifie l'interrogation des bases de connaissances, permettant de déterminer, par exemple, les équivalences parmi les services et les appareils connus, les correspondances entre les composants de Patterns des deux catalogues spécifiques Agnostic et Vendor ou des informations sur le paramètre d'interface d'un service Cloud spécifique ou d'un appareil [99].

III.5.2.2. Représentations formelles de pattern en utilisant (UML) :

UML fournit des concepts de modélisation pour représenter des outils logiciels, des plateformes et des infrastructures à partir de différents points de vue [100]. Par conséquent, fournir des extensions à UML satisfera les exigences actuelles et facilitera la compréhension des modèles et les exigences en matière de modélisation de Cloud. En particulier, lorsque les scénarios de migrations orientées vers le Cloud [101] doivent être pris en charge.

Afin de modéliser le déploiement des applications Cloud qui sont parfaitement applicables aux modèles UML, un langage de modélisation d'application Cloud (CAML) a été proposé dont l'objectif est d'exprimer les topologies de déploiement par des concepts de modélisation de Cloud communs et pour permettre le câblage de ces modèles avec les offres concrets des fournisseurs de Cloud. Ce câblage est réalisé en appliquant un profil CAML dédié à un modèle de déploiement exprimé en termes de bibliothèque CAML [100].

III.5.2.3. Représentations formelles de pattern en utilisant Workflow :

De nombreuses applications Cloud nécessitent l'achèvement de multiples tâches interdépendantes, la description d'une activité complexe impliquant un tel ensemble de tâches est connue en tant que flux de travail [102].

Selon [103], les aspects les plus importants qui différencient un système de flux de travail Cloud du système conventionnel est l'activité orientée vers le marché. En outre, les auteurs affirment que le rôle d'un système de flux de travail Cloud est de faciliter l'automatisation des applications de flux de travail soumis aux utilisateurs où les tâches ont des relations de priorité définies par des outils de modélisation basés sur des graphiques tels que DAG (Graphique acyclique dirigé) et réseaux de Petri ou outils de modélisation basés sur les langages tels que XPDL (XML Process Definition Language).

III.5.2.3.1. Petri net (ICNets) :

Comme leur nom indique, les IC (Inter-Cloud) sont basés sur des réseaux de Petri. Les réseaux de Petri sont des formalismes de modélisation simples et graphiques avec une base mathématique solide, leur efficacité pour la modélisation du flux de travail n'a pas besoin d'être prouvée [104].

Afin d'illustrer les caractéristiques des ICNets (Inter-Cloud Workflow Management Systems by Petri Nets), certains flux de travail traitant du stockage dans les Cloud ont été implémentés. Ces flux de travail consistent à effectuer certains transferts de données et une gestion dans OpenStack du Cloud [102]. La (figure 7) montre certaines opérations modélisées par les ICNets.

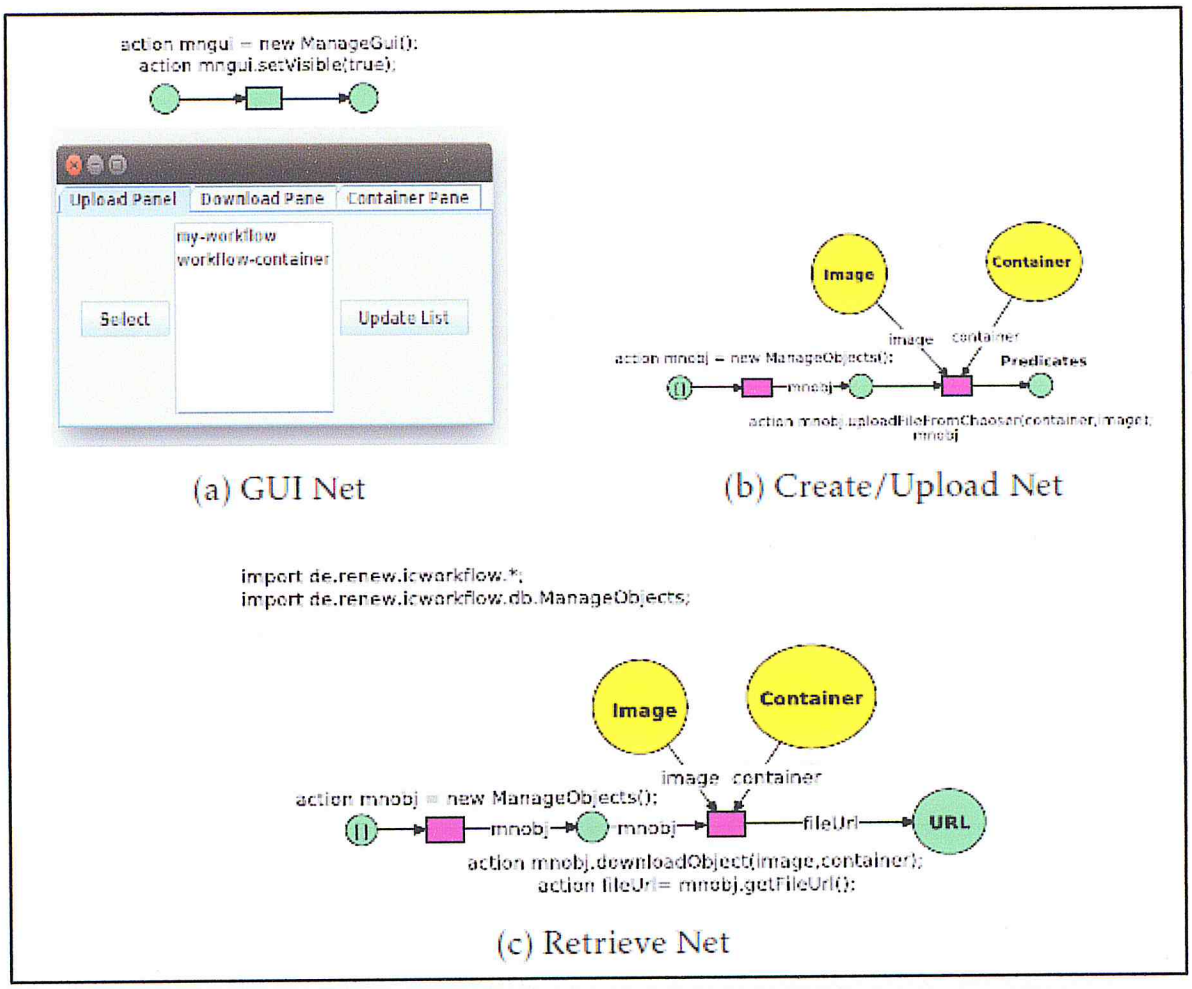


Figure 7 : Opérations Cloud avec ICNets [102]

III.5.2.4. Langage patterns de formalisation de formes (OWL, OWL-S) :

Langage complet de formalisation patterns, basé sur les ontologies OWL [105] et OWL-S, pour décrire à la fois des designs et des modèles Cloud. L'utilisation combinée des ontologies OWL et des descriptions de services Web sémantiques assure une représentation facilement extensible des patterns, dont les participants peuvent être directement connectés aux composants logiciels, à des services Web ou à des Cloud qui les mettront en œuvre. En outre, les ontologies réduisent les problèmes liés aux modèles et les actifs logiciels sont décrits par des personnes ayant des arrière-plans.

III.5.3. Model-Driven Engineering Approaches (MDA):

L'architecture axée sur le modèle OMG (MDA) est une approche basée sur un modèle pour le développement de systèmes logiciels. Les principaux avantages du MDA sont la simplification de la portabilité, de l'interopérabilité et de la réutilisation

des parties du système qui peuvent être facilement déplacées d'une plate-forme vers une autre, ainsi que la maintenance du système grâce à une lecture lisible par l'utilisateur et des spécifications réutilisables à différents niveaux d'abstraction. Dans le contexte du Cloud Computing, le développement axé sur les modèles peut être utile pour permettre aux développeurs de concevoir un système logiciel dans un Cloud et d'être soutenu par des techniques de transformation de modèles dans le processus d'instanciation du système Cloud spécifiques et multiples [92]. Pour cette raison, la combinaison de l'ingénierie des applications axée sur les modèles et du domaine du Cloud Computing est actuellement au centre de plusieurs groupes et projets de recherche : approche axée sur le modèle pour la conception et l'exécution d'applications sur plusieurs Cloud (MODACLOUDS)[106], Advanced Pro-Based Service Based Software and Migration of Legacy Software (ARTIST) [107], Cloudware de plate-forme Cloud Platform (PaaSage) [108], InterCloud Architecture (ICA)[109].

III.5.4. Systèmes multi-agents (MASs) :

Un MAS peut être décrit comme un système informatisé composé d'agents intelligents interactifs, collaborant dans le même environnement [92].

La communication dans les systèmes multi-agents est à la base des interactions et de l'organisation des agents mais aussi de la résolution coopérative des problèmes [110]. Elle permet de synchroniser les actions des agents et de résoudre les conflits de ressources et de buts par la négociation. En effet, sans communication, les agents sont incapables de coopérer, de négocier, de coordonner leurs actions ou de réaliser des taches en commun. Deux modes de communication se distinguent dans la littérature : la communication indirecte qui se fait par transmission de signaux via l'environnement et la communication directe qui correspond aux échanges des messages entre les agents. L'utilisation de multi-agent rend le Cloud à meilleur service [111].

III.5.5. Moteur sémantique (semantic engine) :

Le moteur sémantique effectue des tâches pour classer les ressources Cloud similaires avant de fournir des services Cloud sémantique. Il calcule la ressemblance entre les ressources Cloud en utilisant une analyse statistique, c'est une analyse de cluster, qui est basée sur la spécification de ressources et d'une application [112].

Lorsque le calcul se produit, le moteur sémantique considère les poids des ressources en fonction des spécifications d'application, il fait des groupes, qui ont des informations sur les ressources similaires, avec des résultats en comparant les ressources. Par conséquent, selon les spécifications de l'application, les ressources Cloud similaires sont classées dynamiquement en tant que groupes comme le montre la (figure 8) [112].

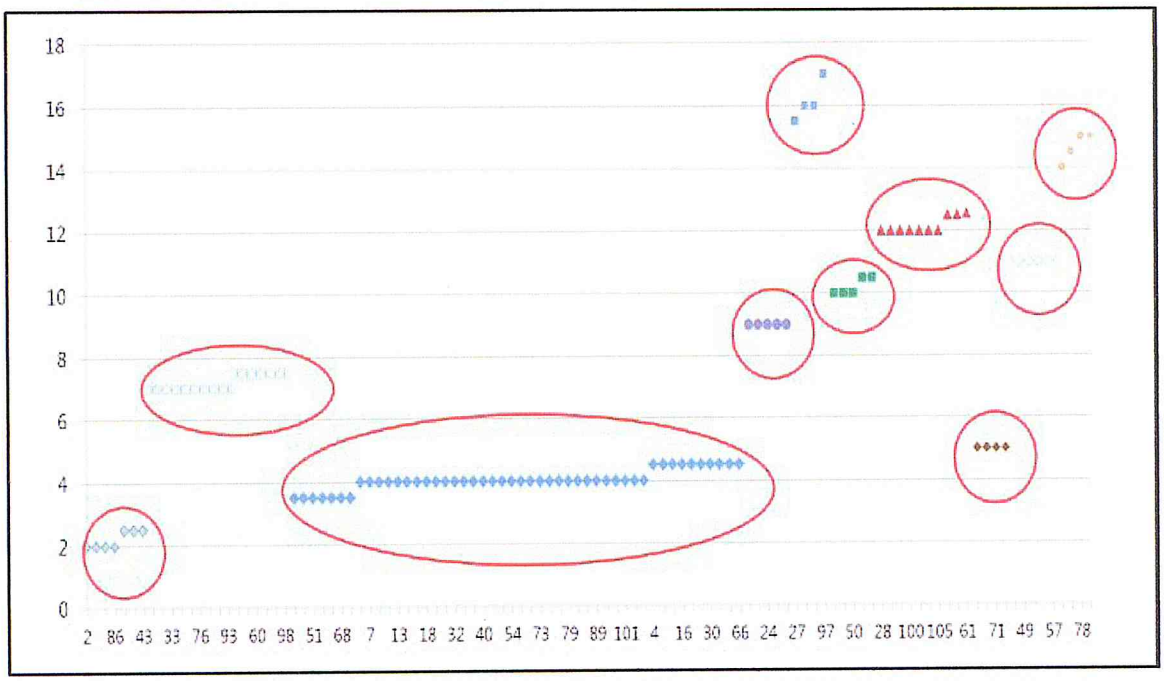


Figure 8 : Le résultat de l'analyse de classification [112]

III.6. Expériences de réalisation de l'interopérabilité entre Clouds :

Les premiers efforts pour la portée, l'étude et l'adressage de l'interopérabilité sémantique entre les Cloud qui échangent leurs informations sont déjà apparues. Les conflits d'interopérabilité sémantique surviennent, en particulier dans le cas d'un Cloud hybrides, tel le cas de la collaboration des services publics et privés afin de fournir des solutions plus sophistiquées et à valeur ajoutée, ou le cas d'un courtier Cloud responsable de la coordination et du service d'approvisionnement selon les besoins d'un utilisateur spécifique [91].

Cette section présente les cadres d'interopérabilité sémantique du Cloud Computing qui définissent les principales orientations pour l'investigation et

l'avancement de la sémantique et contribuent à résoudre les problèmes d'interopérabilité sémantique qui existent dans les infrastructures Cloud actuelles.

III.6.1. mOSAIC :

En 2010, la Commission européenne a lancé le projet mOSAIC [113]. L'objectif principal de ce projet est de créer et de développer une plate-forme open source capable de créer un pont entre les services Cloud et les applications et les relier entre eux. Les concepteurs et les développeurs de cette plate-forme espèrent pouvoir émettre plus de concurrence entre les fournisseurs de Cloud à l'avenir. Cette plate-forme permettra aux applications de recevoir des besoins de service que les utilisateurs demandent, puis d'envoyer les modifications ou les besoins proposés sur la plate-forme via une API constante [114]. Ensuite, la plate-forme recherchera des services compatibles avec les demandes et les besoins des utilisateurs et expédiera l'information résultante sur la plate-forme [113].

III.6.2. CONTRAIL :

Le projet CONTRAIL [113] financé par l'union européenne vise à créer un environnement dans lequel chaque institution pourrait être à la fois un fournisseur de Cloud et un client. Selon les spécifications de ce projet, une organisation peut devenir un fournisseur de Cloud dans le cas où son infrastructure n'est pas utilisée à fond et elle peut devenir un client lorsqu'elle sera confrontée aux heures les plus fréquentées [113].

Comme il est montré à la (figure 9), une interface standard sera créée pour mener le partage des ressources et la collaboration entre les fédérations de Cloud, en conséquence, un système open source peut être créé, déployé, évalué et développé où les ressources de différents opérateurs peuvent être combinées dans un seul Cloud fédéré homogène [114], ce type de Cloud peut être consulté de manière constante par les utilisateurs.

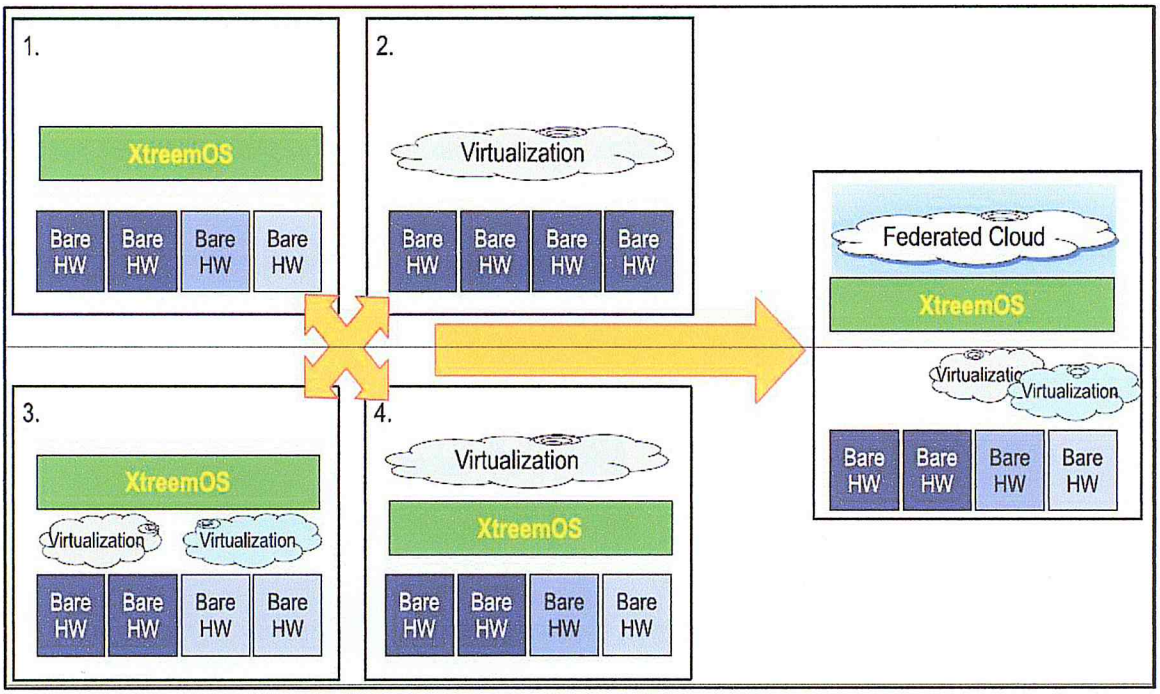


Figure 9 : intégration de plusieurs Cloud indépendants dans un Cloud fédéré [113]

III.6.3. Vision Cloud :

Le Cloud VISION vise à introduire l'architecture pour une infrastructure orientée Cloud (Figure 10) [113]. Cette architecture est appliquée pour concevoir un déploiement de référence pour les nouvelles technologies et les standards ouverts, par la suite, un schéma ou cadre apparaîtra qui est en mesure d'offrir des services de stockage centrés sur les données de manière adaptative, d'où le problème de verrouillage des données peut être facilement résolu et une interopérabilité sécurisée ou sure des données sera garantie [113].

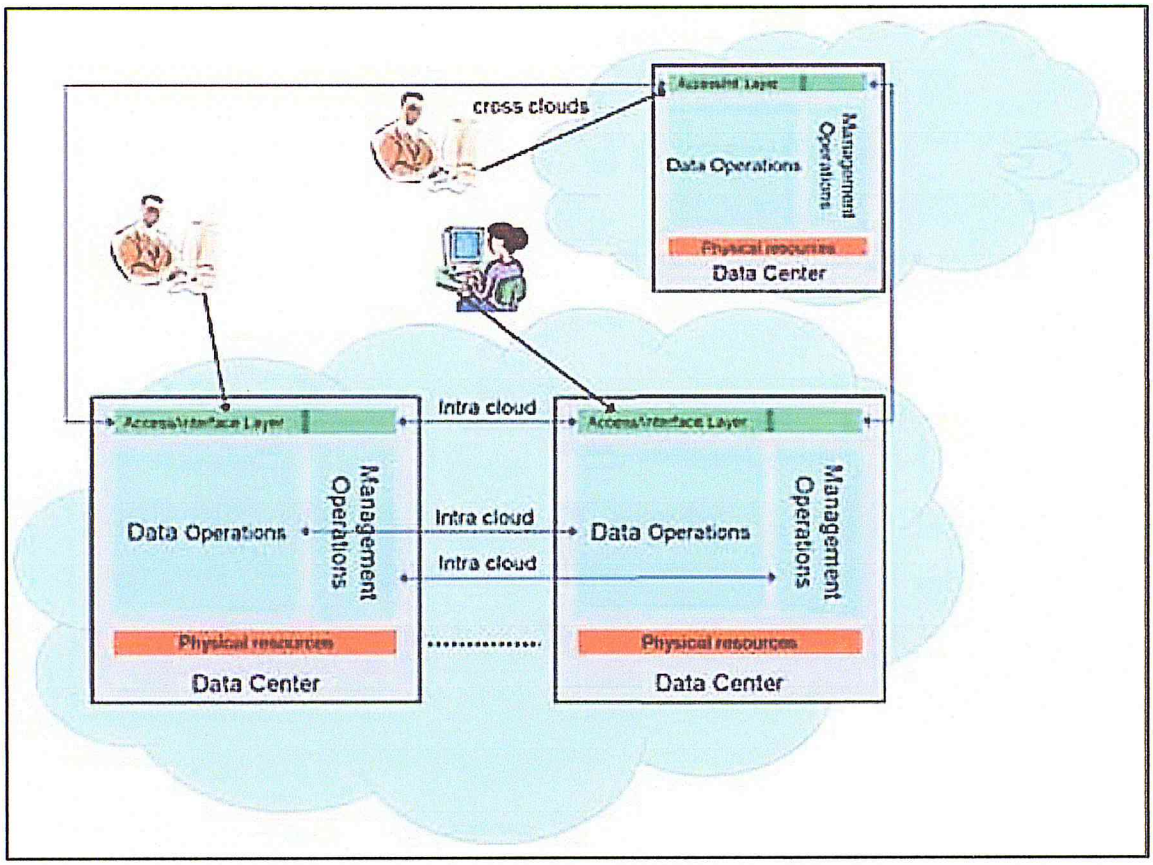


Figure 10 : Infrastructure Vision Cloud [113]

III.6.4. REMICS :

L'objectif de REMICS est de développer une méthodologie et des outils axés sur les modèles avancés pour la réutilisation et la migration des applications existantes vers des services Cloud interopérables [114]. Le paradigme du Cloud de service est synonyme de combinaison de Cloud Computing et d'architecture orientée services (SOA) pour le développement de systèmes logiciel en tant que service (SaaS) [113;115]. Il est nécessaire de gérer des modèles particuliers d'architecture et de méthodes basées sur des modèles afin de transférer l'architecture et de superviser les caractéristiques spécifiques du cadre de développement des services Cloud (Figure 11) [113].

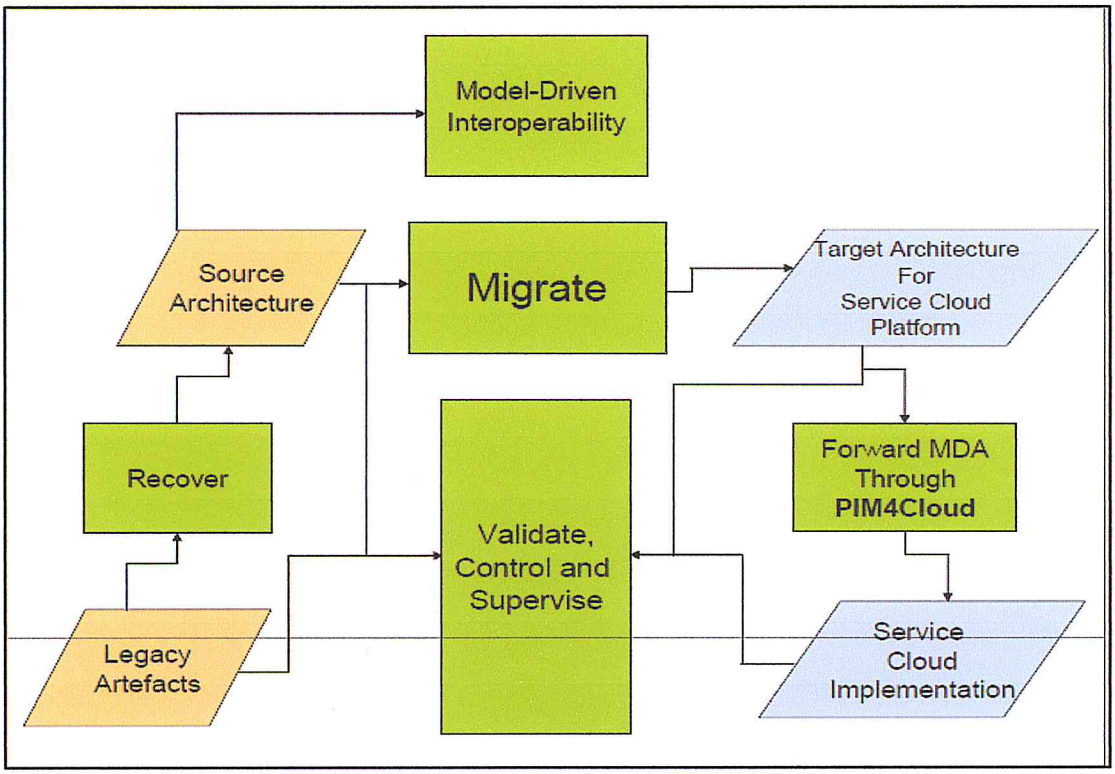


Figure 11 : Vue d'ensemble de REMICS [113]

III.6.5. RESERVOIR :

L'objectif majeur du projet RESERVOIR est de créer une infrastructure axée sur les services novateurs dont l'accent est mis sur des services. Cette infrastructure renforcera l'interopérabilité vigoureuse entre les fournisseurs de Cloud pour offrir des services en tant que ressources de manière fiable, ainsi, différents utilitaires informatiques avec des technologies indépendantes et distinctes peuvent être créés. Ces services publics peuvent être accessibles à chaque fois que sa serra nécessaires et ils apportent plus de sens à la compétitivité dans l'économie de l'UE [114].

Dans le projet RESERVOIR, l'interopérabilité s'exerce dans toutes les couches architecturales. L'architecture de ce projet dicte que les fournisseurs de Cloud doivent exprimer les demandes dans une langue similaire, par conséquent, cela peut entraîner l'interopérabilité entre eux. De ce fait, il sera possible aux fournisseurs de services de sélectionner leurs fournisseurs de sites RESERVOIR préférés en fonction de leurs besoins, du cout, leurs activités seront plus tangibles sur le marché de fourniture du Cloud Computing [116]. La (figure 12) décrit les composants d'une architecture RESERVOIR.

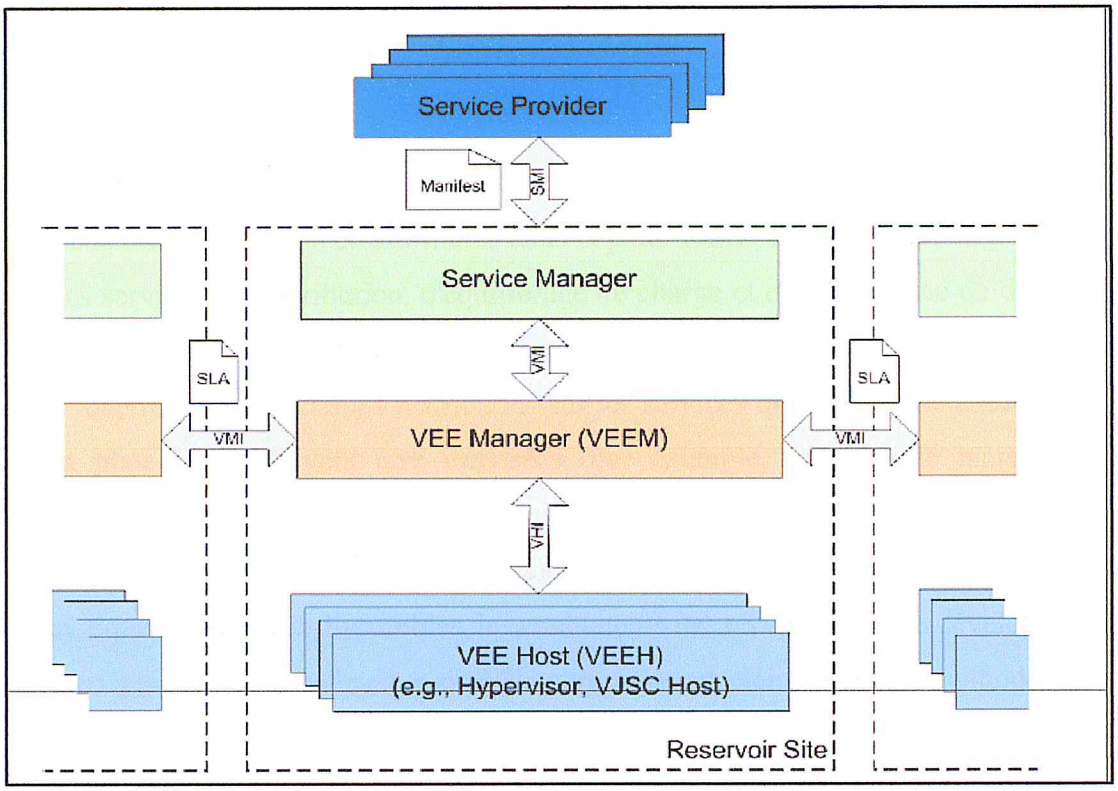


Figure 12 : Architecture proposée par le projet RESERVOIR [116]

III.6.6. SITIO :

SITIO se concentre sur la création d'une architecture qui assure l'accès des courtiers aux services métier et au Cloud Computing avec un coût minimum. Sur la base de ce dernier, une plate-forme sera développée qui servira comme un moyen d'assurer une interopérabilité fiable, sécurisée et rentable entre des applications hétérogènes, grâce à la combinaison de concepts tels que le logiciel en tant que service, la sémantique, la modélisation des processus métier et le Cloud Computing [117]. La (figure 13) illustre les éléments de base de l'architecture du projet.

Quatre éléments de base sont identifiés dans l'architecture SITIO, à savoir l'interface utilisateur, les services de processus, les services aux entreprises et les services de métadonnées. L'interface utilisateur permet à différents utilisateurs d'accéder à la plate-forme SITIO et d'utiliser et de gérer leurs comptes et leurs applications, les développeurs de logiciels et les fournisseurs visitent l'interface afin de déployer leurs applications, tandis que les utilisateurs finaux peuvent effectuer une recherche dans le référentiel des applications et s'abonner à ceux qui leur intéressent [114].

d'exigences [118]. Son objectif principal est de combiner des innovations sélectionnées dans le domaine des architectures et technologies orientées services pour faciliter la mise en œuvre des environnements de service interopérables [118].

III.6.8. Cloud@Home :

L'objectif de ce projet est de réaliser et créer une infrastructure Cloud capable de distribuer des services et des ressources au sein d'un Cloud de manière appropriée [114]. Aussi, ce projet vise à créer des Cloud commerciaux qui peuvent être utilisés comme base d'un marché ouvert. Dans ce genre de marché, les utilisateurs sont autorisés à négocier ou à échanger des services de manière payante par chaque utilisation [119].

La (figure 14) illustre l'architecture de Cloud@Home, la responsabilité de la couche frontière est de gérer les services et les ressources grâce à une approche universelle du système Cloud, les besoins des utilisateurs finaux sont converties en demandes de ressources physiques via cette couche [114].

D'autres fonctions sont également effectuées dans cette couche, y compris la surveillance, la négociation et l'ajustement des SLA dans les Cloud commerciaux. Par conséquent, l'interopérabilité peut être obtenue entre les Cloud.

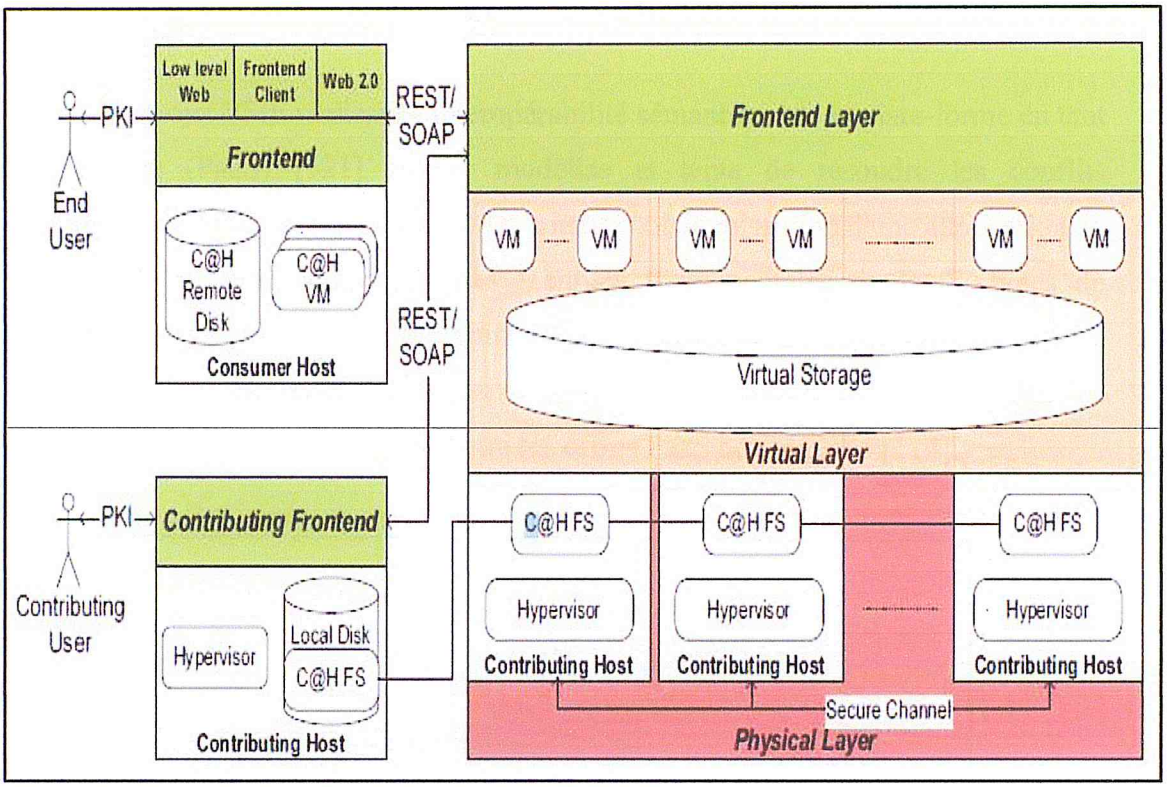


Figure 14 : Architecture de base du projet Cloud@Home [119]

III.6.9. SOA4All :

L'objectif du projet SOA4All est de créer un environnement dans lequel les services peuvent être diffusés et appliqués par de nombreuses parties. Les concepteurs et les développeurs de ce projet tentent de créer une structure qui peut combiner les innovations techniques complémentaires et évolutives en une plate-forme de provisionnement de services avec un domaine cohérent et indépendant [120].

SOA4All est capable de construire, d'offrir, d'utiliser et d'analyser les services qui sont distribués dans SOA4All. Il existe trois sous-composants dans cette architecture qui sont chargés d'effectuer trois tâches différentes liées à la gestion des services, ces trois tâches comprennent l'analyse en temps d'exécution, le provisionnement au moment du design et de la consommation.

L'architecture de SOA4All repose sur une pierre angulaire de l'infrastructure appelée bus de service distribué. Il connecte et intègre tous les composants SOA4All et les faits de coopérer entre eux en intégrant les espaces sémantiques et le bus de service [114].

permettre aux entreprises européennes de prendre une position de leader mondial [123] [124].

Dans PaaSage, les partenaires industriels et académiques travaillent en étroite collaboration avec les partenaires fournisseurs Cloud pour rechercher, développer et déployer des applications à forte intensité de ressources qui bénéficient grandement de la gestion basée sur les modèles et exploitent l'évolutivité selon les exigences personnalisées [125].

III.6.12. 4CaaSt :

Le projet 4CaaSt vise à créer une solution pour la création, le marketing, le déploiement et la gestion des applications dans le Cloud, à la fois sur les produits de plate-forme et la plate-forme en tant que service (PaaS). 4CaaSt présente le concept de Blueprint, une description technique d'une application ou d'un service qui découple les différentes dépendances qu'il possède tout au long des couches Cloud [126].

Grâce à Blueprint et à la manière dont les applications et les services sont négociés, provisionnés et déployés, 4CaaSt peut prendre en charge plusieurs modèles d'utilisation et d'affaires offrant aux logiciels et aux fournisseurs de services la flexibilité pour utiliser les ressources et les services qu'ils préfèrent [126].

En résumé, 4CaaSt permet aux fournisseurs de logiciels et de services de se concentrer sur leur entreprise (le logiciel et la monétisation des services), laissant la complexité sous-jacente de l'infrastructure et des plates-formes hors leurs préoccupations. À ce jour, une première version de la plate-forme a été livrée, implémentant les principales fonctionnalités pour l'exécution [126].

III.6.13. Cloud-TM :

Cloud-TM est explicitement adapté pour correspondre aux exigences d'applications du Cloud Computing, il vise à étendre le modèle de programmation impératif des systèmes DTM qui utilisent l'abstraction non seulement comme moyen de simplifier la programmation en parallèle mais aussi comme moyen naturel d'agréger l'efficacité de la communication en évitant les pièges de performance propres à DSM sans sacrifier la simplicité de programmation[127].

En réalité, contrairement aux DSM fortement compatibles, qui nécessitent une télécommande de synchronisations coûteuse à chaque accès unique au mémoire, les

transactions atomiques permettent des implémentations optimistes qui permettent de lier toute action de cohérence au sein d'une seule synchronisation la phase à prendre en compte [128, 129, 130]. Cette approche amortit les frais généraux de communication sur un (éventuellement grand nombre d'accès mémoire), avec des avantages clairs en termes de performance [127].

III.6.14. L'architecture Cloud Computing orientée service (SOCCA) :

L'architecture Cloud Computing orientée service (SOCCA) combine les technologies SOA et Cloud Computing afin d'améliorer l'interopérabilité entre Clouds. La couche individuel\fournisseur du Cloud est la partie inférieure où chaque fournisseur développe ses propres centres de données et fournit ses services. L'innovation des ressources de la SOCCA réside dans le regroupement de ces ressources en un ensemble de services indépendants et accessibles par des interfaces ouvertes et standardisées, à partir de là, ils peuvent être combinés avec des services d'autres fournisseurs Cloud [131].

La couche de cartographie de l'ontologie Cloud est utilisée dans le cas où des fonctionnalités supplémentaires, non incluses dans les normes, doivent être mis en œuvre. Le Cloud Broker Layer sert d'agent entre les fournisseurs Cloud individuels et la couche SOA qui est responsable de l'édition des services [131].

III.6.15. Cloudbus InterCloud :

Buyya et al. [132] présentent l'idée d'un Cloud fédéré dans l'environnement informatique qui facilite l'évolutivité et la fourniture de services dans des conditions variables [133]. Les principaux éléments de l'architecture fédérée est le courtage ainsi que le coordonnateur des clients prestataires de services [91]. Un client lance un courtier Cloud afin de rencontrer ses besoins, tandis que les coordonnateurs de Cloud publient leurs services à la fédération. Cloud exchange agit comme un médiateur réunissant les fournisseurs de services et les clients, il agrège les demandes d'infrastructure à partir des courtiers d'applications et il les évalue en fournissant les ressources disponibles publiées par le coordonnateur.

La (figure 17) présente un exemple d'un modèle sémantique orienté système, ce type de configuration peut également être lié à la DSL via des annotations.

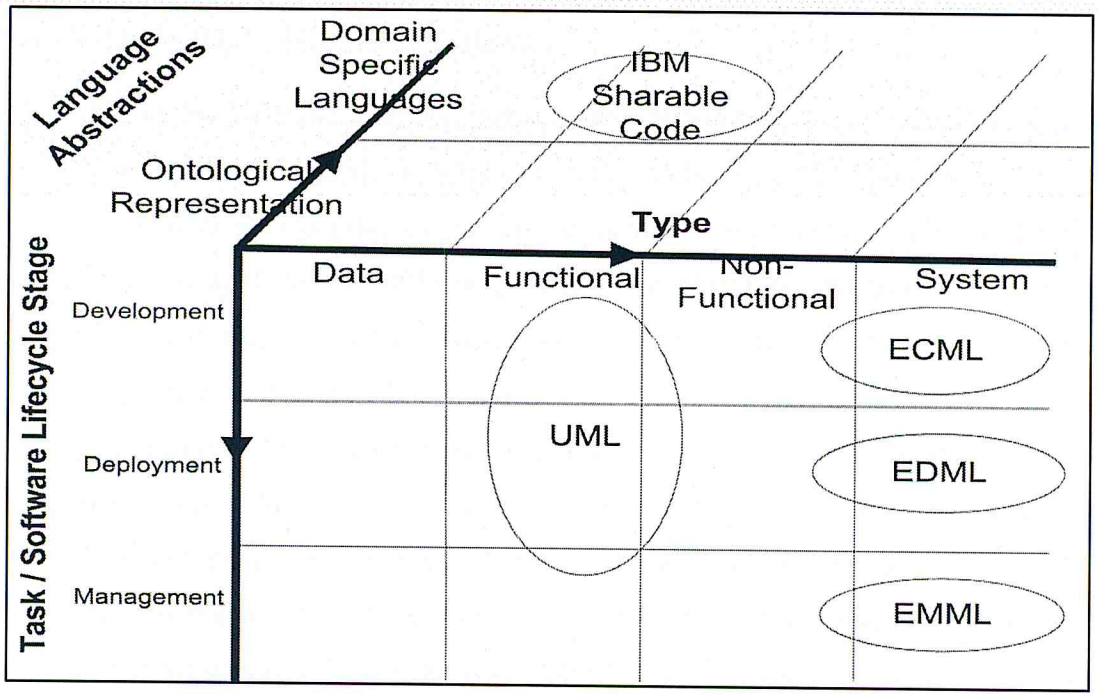


Figure 17 : Tranchage dimensionnel de l'espace de modélisation [134]

III.7.2. L'OWL-S de CMU (CODE) :

CODE [136] prend en charge la totalité des processus de développement des services Web de la génération de description des services jusqu'au déploiement et l'enregistrement d'un service. CODE est implémenté sous la forme d'un plug-in Eclipse prenant en charge des activités pour les fournisseurs SaaS et les développeurs de systèmes grand public SaaS. En plus des outils pour la description des services, CODE comprend les éléments Matchmaker et la machine virtuelle (VM) [114].

La couche d'interopérabilité sémantique fournit un fichier de description syntaxique au service générateur de description sémantique pour générer une description sémantique aux fichiers de service.

Grace à l'utilisation du composant WSDL2OWL-S inclus dans CODE, le profil du service est généré pour l'interface Web service. En utilisant l'éditeur OWL-S, l'éditeur de description sémantique de service peut ajouter des détails tels que les transformations de données spéciaux dans le service fondation, flux de contrôle et

informations de flux de données dans l'élément du modèle de procédure de service (processus) et paramètres non fonctionnels (par exemple, évaluation de qualité) dans l'élément du plan de service (profil) [114].

III.7.3. SLA@SOI :

SLA@SOI est un projet financé par l'UE, envisage une infrastructure axée sur les services et axée sur l'entreprise en assurant un service économique souple et fiable. Le projet tente de créer un cadre où les services Cloud peuvent être négociés en biens économiques et les accords SLA peuvent être établi entre les clients et les services/ entreprises, les fournisseurs de services et les fournisseurs d'infrastructure. Ce cadre de gestion multi-domaine SLA exige le suivi du cycle de vie des services [137].

SLA@SOI a apporté des contributions significatives à l'organisme de normalisation tel, Open Cloud Grid Forum (OGF), la norme Open Cloud d'interface informatique (OCCI) a été co-présidée par SLA@SOI afin de créer une interface générique pour l'infrastructure Cloud Computing à travers laquelle les paramètres SLA peuvent communiquer [130]. Les chercheurs de SLA@SOI ont également contribué à un module SLA dédié à l'Unified Service Description Language (USDL), une norme proposée au W3C [130].

III.8. Discussions :

Les initiatives de normalisation et les modèles d'interopérabilité examinés dans la section précédente utilisent des stratégies similaires pour traiter les problèmes d'interopérabilité entre Clouds qui surviennent lorsque différents fournisseurs de Cloud tentent de coopérer pour échanger des données, des applications et des machines virtuelles. Ces incompatibilités peuvent être, par exemple techniques, des implémentations de virtualisation incompatibles (VMware, Xen et KVM) ou un code de programmation incompatible (basé sur Java, basé sur PHP) ou sémantique, par exemple, différents fournisseurs Cloud utilisent des modélisations et des notations différentes pour exposer les mêmes caractéristiques [76].

Afin de réaliser une étude comparative des standards existants pour l'interopérabilité des services Cloud nous avons construit le (tableau 3) suivant :

Standard	Niveau
Open-SCA [84]	SaaS
USDL [85]	SaaS
EMML [86]	SaaS
OCCI [87]	PaaS
CIMI [88]	PaaS
CAMP [89]	PaaS
TOSCA [88]	PaaS
CDMI [88]	PaaS
OVF [90]	IaaS

Tableau 3 : Les efforts de normalisation de l'interopérabilité dans le Cloud Computing

A partir du tableau construit, nous avons remarqué qu'il y avait un manque de standards pour l'interopérabilité des services Cloud de type SaaS, et que la pluparts des standards existants ont été conçu pour les services Cloud de type PaaS.

Suite aux standards nous avons construit le (tableau 4) suivant qui classe l'ensemble des modèles réalisant l'interopérabilité sémantique dans l'environnement du Cloud Computing.

Solution	Niveau	Modèle et compositions de l'architecture	Ontologie
mOSAIC [113][114]	IaaS	Mécanisme de courtage multi-agent, exécuteur d'application, courtier de ressources	Oui
CONTRAIL [113][114]	IaaS/PaaS	Système open source	Non
Vision Cloud [113]	IaaS	Data center	Non
REMICS [113][114][115]	SaaS	Architecture orienté service (SOA), Modèle @Runtime, Modèle PIM4Cloud	Non

RESERVOIR [114][116]	IaaS	Virtual Execution Environment Host (VEEH), Virtual Execution Environment Manager (VEEM) et le gestionnaire de service	Oui
SITIO [114][117]	SaaS	Interface utilisateur, les services de processus, les services aux entreprises et les services de métadonnées.	Non
NEXOF [118]	-	Entités clés de modèle de référence, l'ensemble pattern, catalogue des normes, catalogue de composants	Oui
Cloud@Home [114][119]	IaaS	Interface web, interface utilisateur Web 2.0, client Cloud@Home (avec requête REST ou SOAP)	Non
SOA4All [114][120]	-	Service web	Oui
PSIF [114][121]	PaaS	Entités fondamentales de PaaS	Oui
PaaSage [122] [123][124][125]	IaaS/PaaS	Unités logiques ou modules, la programmation orientée objet et Workflow composition	Oui

Tableau 4 : Les modèles d'interopérabilité sémantique entre Clouds

Apres avoir passé par étude comparative des modèles existants pour l'interopérabilité des services Cloud, qui a été représentée dans notre tableau nous avons remarqué que la plupart des solutions existantes mettent l'accent sur l'infrastructure en tant que services et plateforme en tant que services, tandis que la recherche de l'interopérabilité pour les services Cloud de type software as service n'est encore qu'a ses débuts, ainsi nous avons remarqué que pour les deux modèles existants SITIO et REMICS qui ont été conçu pour les services Cloud de type SaaS, ne traitent pas le coté sémantique de l'interopérabilité, et cela a été constaté après le passage par une étude détaillée de l'architecture des deux modèles, nous avons remarqué qu'ils n'ont pas impliqué une ontologie, cela veut dire qu'ils n'ont pas marqué le côté

sémantique de l'interopérabilité, mais ils se sont concentrés plutôt sur l'interopérabilité syntaxique. Par conséquent fournir une interopérabilité sémantique pour les services Cloud de type logiciel en tant que service est notre préoccupation majeure.

Très peu de modèles et standards de l'interopérabilité existants pour les logiciels en tant que service dans l'environnement du Cloud Computing se concentrent sur l'interopérabilité sémantique, ainsi que la pluparts des conflits d'interopérabilité sémantique sont élevés dans IaaS, PaaS et très peu au niveau SaaS. Foster [138] revendique que dans un proche avenir nous devrions exclure les normes multiples des trois couches, seulement celles qui seront utiles vont finalement survivre.

La nécessité d'un modèle d'interopérabilité sémantique pour les logiciels en tant que service dans l'environnement du Cloud Computing est évidente afin de garantir la collaboration et l'interaction par les différents intervenants dans le Cloud. En fait, l'objectif du modèle d'interopérabilité sémantique pour les logiciels en tant que service est de supporter l'échange et la communication entre Clouds ce qui signifie que la solution doit vérifier la négociation des services entre différents fournisseur de Cloud pour résoudre des conflits sémantiques entre des systèmes Cloud hétérogènes.

III.9. Conclusion :

Ce chapitre présente des travaux connexes sur l'interopérabilité sémantique dans le Cloud Computing. Nous avons commencé par l'évolution de l'interopérabilité, ensuite nous avons présenté ses deux dimensions et ses standards. Nous avons fini par examiner les modèles et les solutions existantes d'interopérabilité sémantique, qui ont été organisés en fonction de leurs objectifs.

Un constat sur la littérature montre que la plupart des modèles d'interopérabilité existants mettent l'accent sur l'infrastructure en tant que service et plate-forme en tant que service dans l'environnement du Cloud Computing.

Nous avons conclu que les modèles et les Frameworks d'interopérabilité sémantique pour les logiciels en tant que service sont encore très immatures.

Dans le chapitre suivant, nous présenterons un modèle d'interopérabilité sémantique pour les logiciels en tant que service dans l'environnement du Cloud Computing. Nous allons concevoir notre solution pour aider les logiciels autonomes

en tant que services à interagir et à communiquer entre eux. Le modèle que nous allons proposer serra utilisé comme une interface de connexion en tant que service assurant la communication entre les services Cloud en utilisant les règles de gestion des interactions.

Chapitre IV. *Modélisation de l'interopérabilité sémantique entre SaaS*

IV.1. Introduction :

Après avoir effectué une étude comparative des solutions existantes pour l'interopérabilité des services Cloud dans le chapitre précédent, nous allons dans ce chapitre, présenter notre solution pour l'interopérabilité sémantique des services de type SaaS, l'objectif de ce chapitre est de concevoir un modèle qui pourra faire face au besoin des Clouds tout en assurant les interactions entre eux.

Le modèle s'appuie sur des règles d'interaction sémantiques, l'aspect sémantique nous l'avons introduit dès le début en commençant par construire une ontologie du domaine impliquant des concepts important à la construction des règles d'interactions semi-formelles qui seront par la suite formalisées en utilisant l'algorithme de planification Graphplan.

- **Démarche de travail :**

La démarche de travail que nous avons suivi afin de réaliser notre projet correspond à l'XP-eXtreme Programming [139], qui représente une méthode de développement agile et orientée projet informatique, dont les ressources sont régulièrement actualisées.

C'est une méthode de management de projet destinée à accélérer la réalisation des projets de type flexible. Elle a été conçue à l'origine par Kent Beck et Ron Jeffries pendant le projet "C3" (Chrysler Comprehensive Compensation System) [140].

Tout en mettant l'accent sur les bonnes pratiques de programmation, XP préconise un déroulement par itération courte et géré collectivement, avec une implication constante du client. Il en découle une redéfinition de la relation entre client

et fournisseur, avec de surprenants résultats en termes de qualité de code, de délais et de satisfaction de la demande du client [139].

L'Extreme Programming repose sur quatre valeurs fondamentales [139]:

- La communication :

C'est le moyen fondamental pour éviter les problèmes. Les pratiques que préconise l'XP imposent une communication intense. Les tests, la programmation en binôme et le jeu du planning obligent les développeurs, les décideurs et les clients à communiquer. Si un manque apparait malgré tout, un coach se charge de l'identifier et de remettre ces personnes en contact.

- La simplicité :

La façon la plus simple d'arriver au résultat est la meilleure. Anticiper les extensions futures est une perte de temps. Une application simple sera plus facile à faire évoluer.

- Le feedback :

Le retour d'information est primordial pour le programmeur et le client. Les tests unitaires indiquent si le code fonctionne. Les tests fonctionnels donnent l'avancement du projet. Les livraisons fréquentes permettent de tester les fonctionnalités rapidement.

- Le courage :

Certains changements demandent beaucoup de courage. Il faut parfois changer l'architecture d'un projet, jeter du code pour en produire un meilleur ou essayer une nouvelle technique. Le courage permet de sortir d'une situation inadaptée. C'est difficile, mais la simplicité, le feedback et la communication rendent ces tâches accessibles.

IV.2. Processus de réalisation de l'interopérabilité sémantique des services Cloud :

La (figure 18) illustre la procédure suivie afin de permettre l'interopérabilité sémantique des services Cloud, ce processus comprend cinq étapes :

1. La conversion en PDDL des règles d'interaction semi-formelles introduites.
2. Application du planificateur Graphplan sur le résultat de la première étape.

3. La conversion du plan que Graphplan a obtenu dans l'étape précédente en un ensemble de règles d'interactions.
4. La conversion des règles d'interactions déduites de la planification en langage PDDL.
5. La conversion en OWL des règles d'interaction formelles.

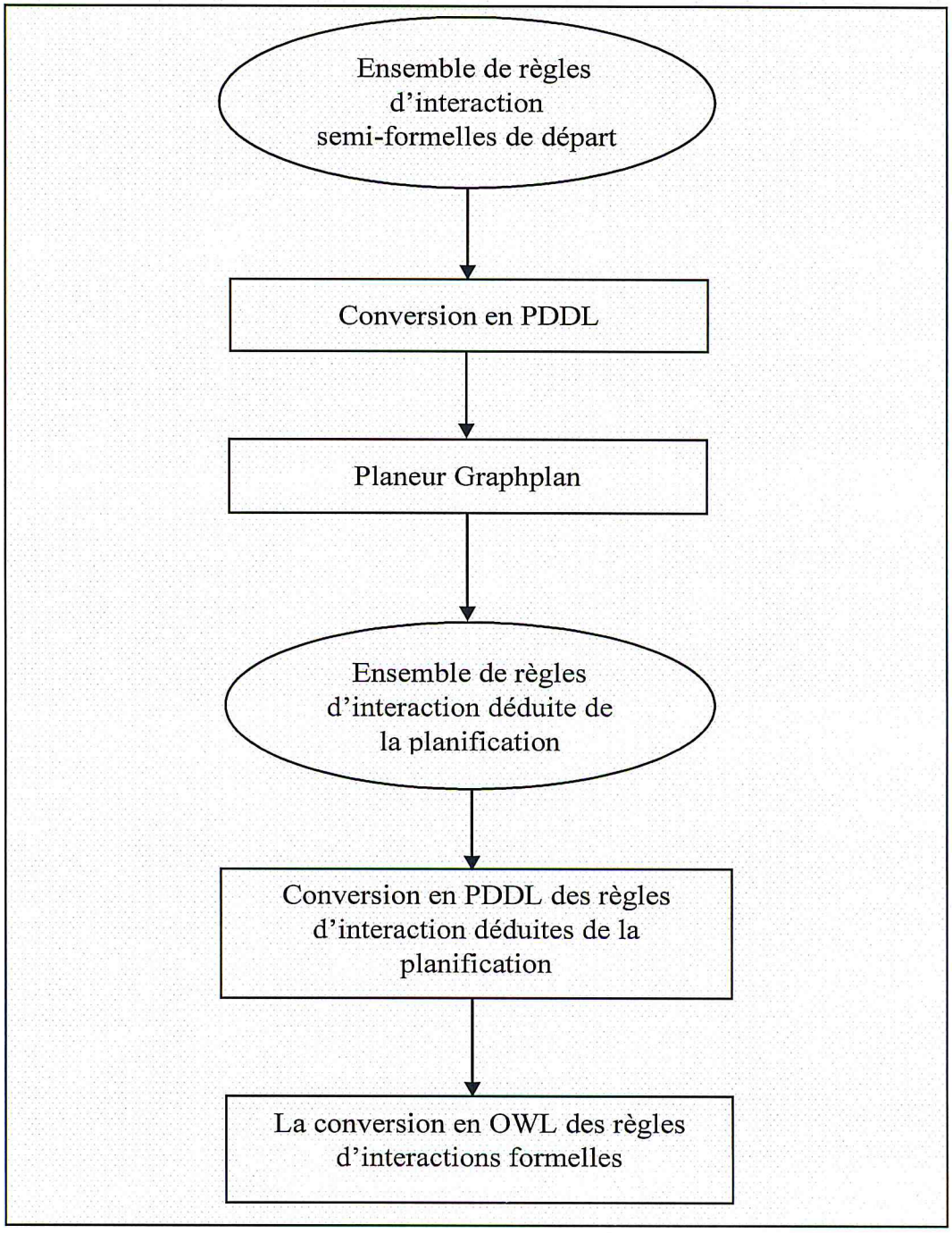


Figure 18 : Processus de réalisation de l'interopérabilité sémantique des services Cloud

Afin de comprendre le fonctionnement de ce processus nous allons l'expliquer étape par étape dans ce qui suit :

IV.2.1. Extraction de règles d'interaction semi-formelles :

Cette étape exige l'extraction des concepts nécessaires à la formulation d'une liste de règles, c'est pour cette raison que nous avons choisi de construire une ontologie rassemblant les termes qui rentrent dans la composition du Cloud et qui peuvent contribuer à la construction des règles d'interactions semi-formelles.

IV.2.1.1. Ontologie :

Une ontologie est un ensemble structuré de concepts permettant de donner un sens aux informations. Les concepts sont organisés dans un graphe dont les relations peuvent être des relations sémantiques, ou des relations de composition et d'héritage.

L'objectif premier d'une ontologie est de modéliser un ensemble de connaissances dans un domaine donné [141].

Dans le domaine du Cloud Computing, les Cloud sont généralement divisés en différents niveaux, comme nous allons démontrer dans notre ontologie (figure 19), qui présente les relations entre les différents composants du Cloud.

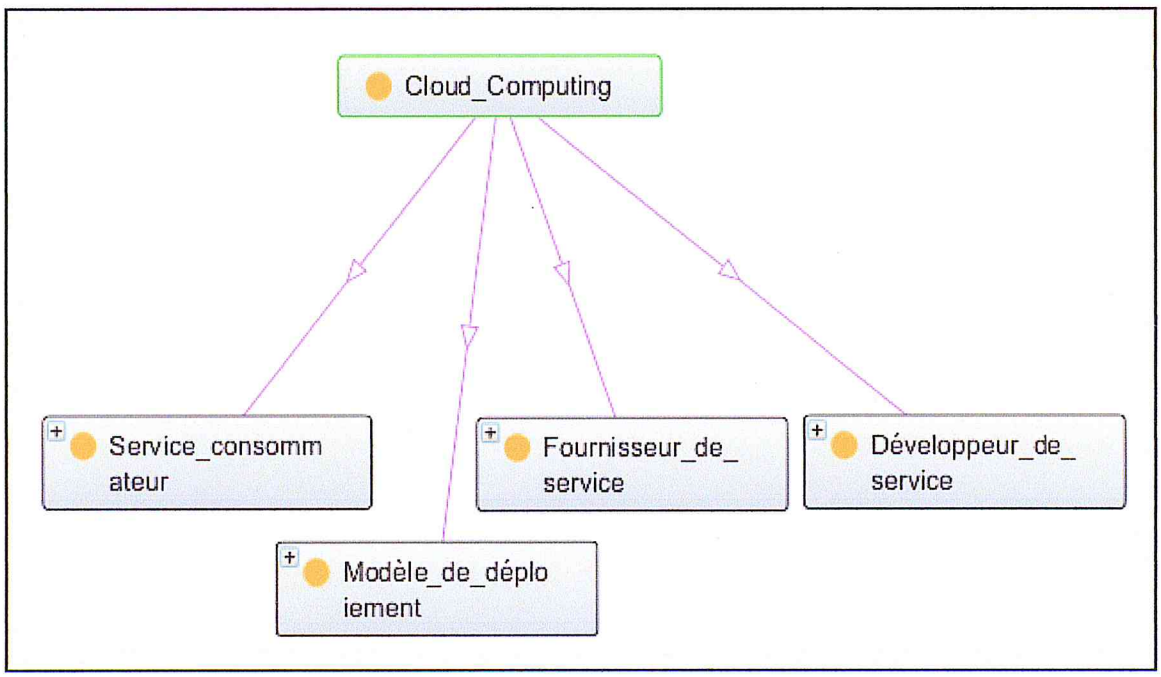


Figure 19 : Ontologie Cloud Computing

➢ **Le consommateur de service :**

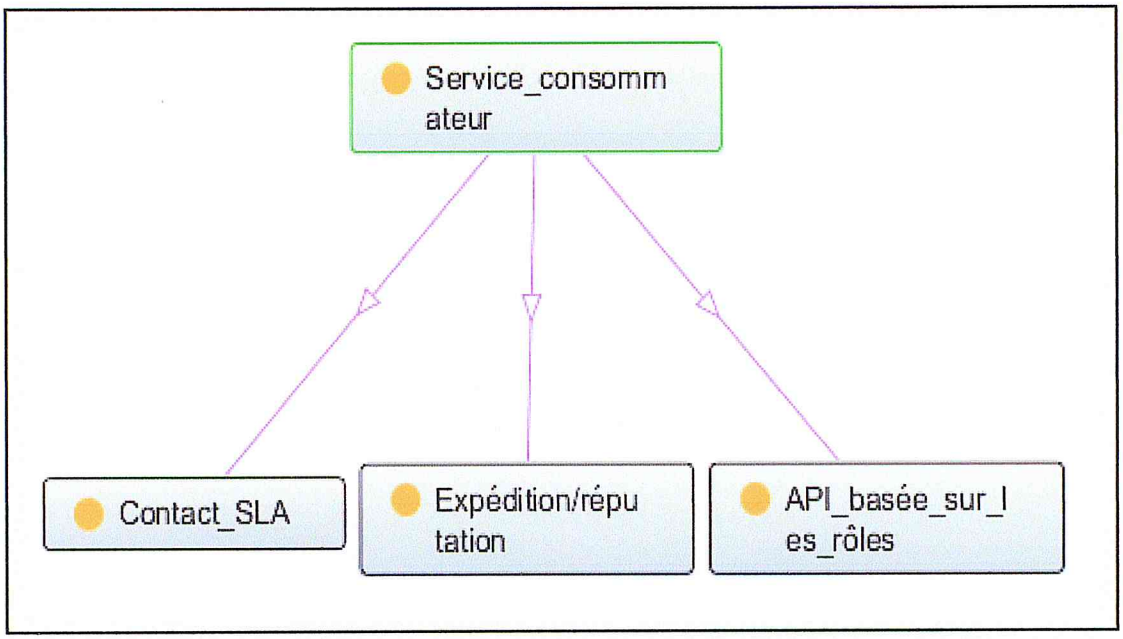


Figure 20 : Ontologie service consommateur

La (figure 20) présente l'ontologie du consommateur de services qui représente l'utilisateur final ou l'entreprise qui utilise réellement les services de tous types soit qu'il s'agisse des logiciels, des plates-formes ou des infrastructures en tant que service [142]. Selon le type de service et son rôle, le consommateur travaille avec différentes interfaces utilisateur et interfaces de programmation. Certaines interfaces utilisateur ressemblent à toute autre application. Le consommateur n'a pas besoin de connaître le Cloud car il utilise l'application [142].

Les autres interfaces utilisateur fournissent des fonctions administratives telles que le démarrage et l'arrêt de machines virtuelles ou la gestion du stockage en Cloud. Les consommateurs qui écrivent le code de l'application utilisent différentes interfaces de programmation en fonction de l'application qu'ils écrivent.

Les consommateurs travaillent également avec des contrats de souscription et des contrats (SLA). En règle générale, ceux-ci sont négociés via une intervention humaine entre le consommateur et le fournisseur.

➢ **Le développeur de service :**

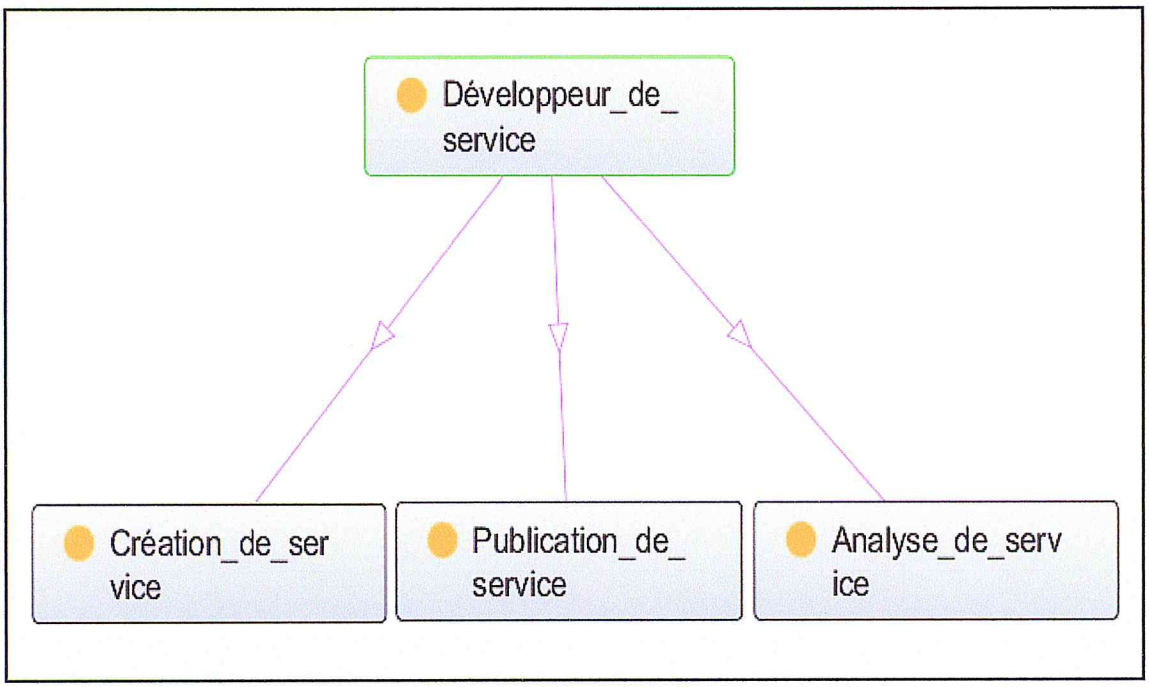


Figure 21 : Ontologie développeur de service

La (figure 21) présente l'ontologie du développeur de services dont son rôle consiste à créer, publier et surveiller les services Cloud. Ce sont généralement des applications «ligne de travail» qui sont livrées directement aux utilisateurs finaux via le modèle SaaS. Les applications écrites aux niveaux IaaS et PaaS seront ensuite utilisées par les développeurs SaaS et les fournisseurs de Cloud.

Les environnements de développement pour la création de services varient. Si les développeurs créent une application SaaS, ils doivent concevoir le code d'écriture d'un environnement hébergé par un fournisseur de Cloud. Dans ce cas, la publication du service le déploie sur l'infrastructure du fournisseur de Cloud [142].

➢ **Le fournisseur de services :**

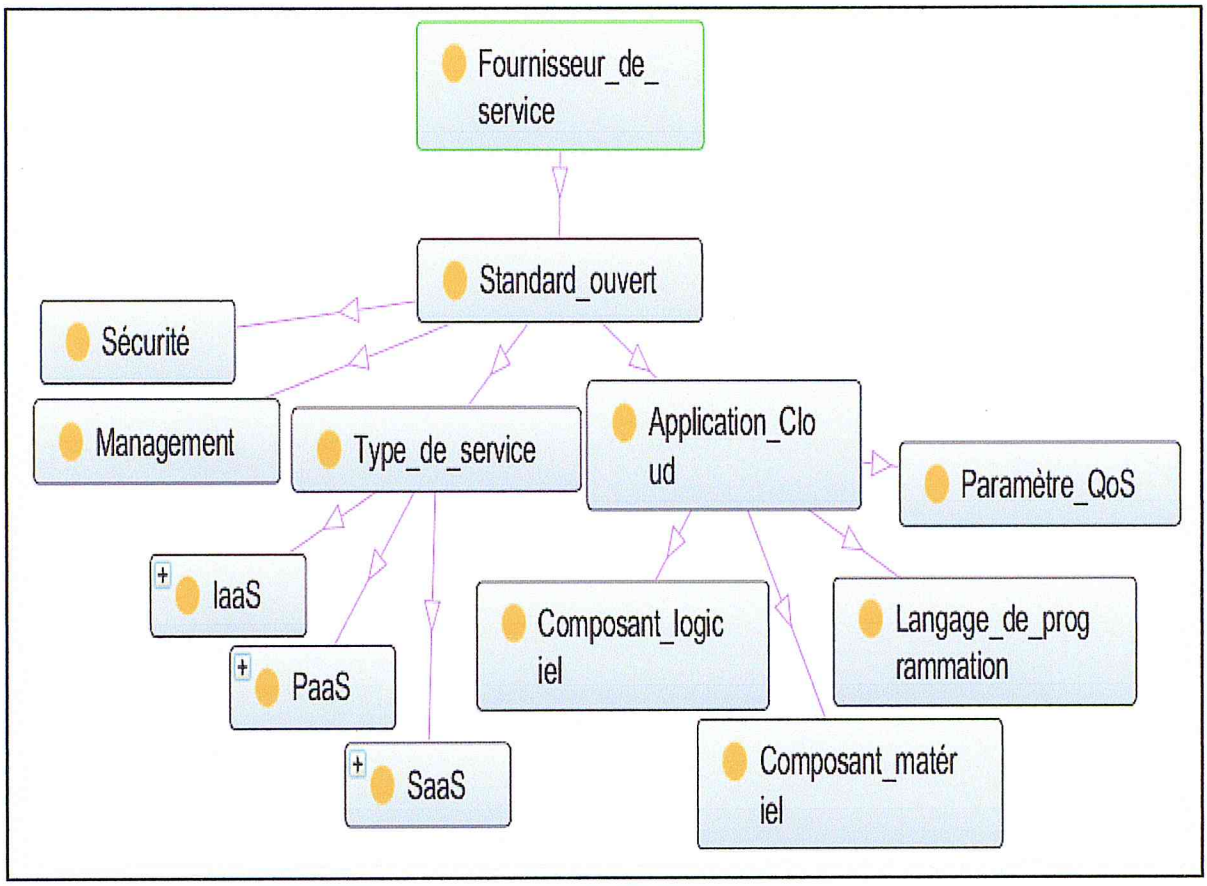


Figure 22 : Ontologie fournisseur de service

La (figure 22) présente l'ontologie du fournisseur de services aux entreprises dont son rôle principale consiste à fournir des services au consommateur. La tâche réelle du fournisseur varie en fonction du type de service [142]:

1. **IaaS :**

Pour l'infrastructure en tant que service, le fournisseur maintient le stockage, la base de données, la file d'attente des messages ou d'autres middleware, ou l'environnement d'hébergement pour les machines virtuelles. Le consommateur utilise ce service comme s'il s'agissait d'un lecteur de disque, d'une base de données, d'une file d'attente ou d'une machine, mais ils ne peuvent pas accéder à l'infrastructure qui l'héberge.

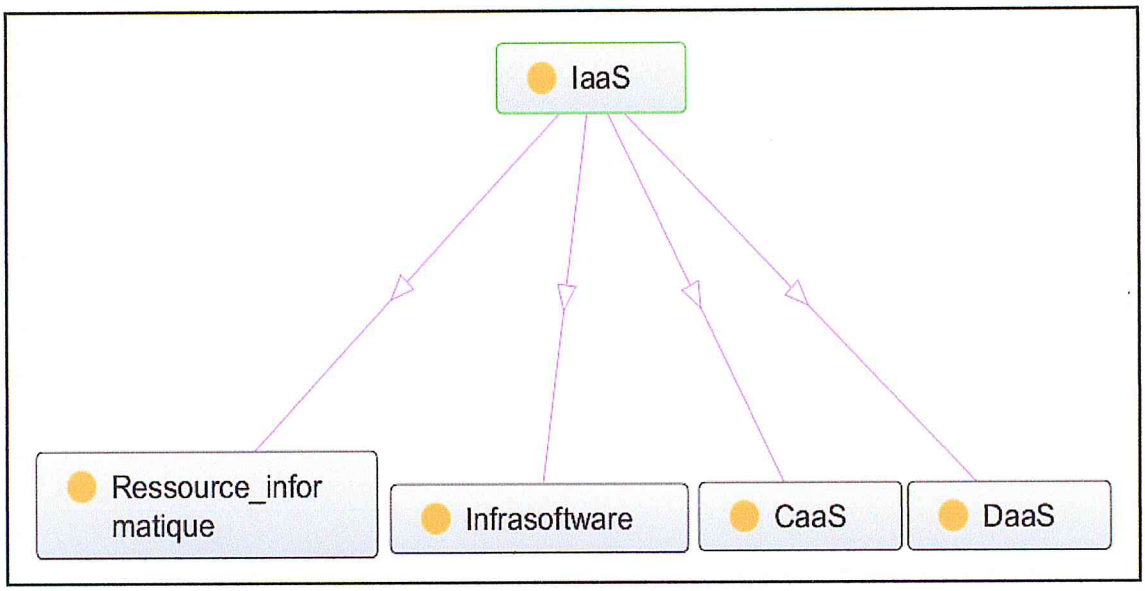


Figure 23 : Ontologie infrastructure en tant que service

Comme il est illustrer dans la (figure 23), l'infrastructure en tant que service (IaaS) prévoit des matériaux, des logiciels et des équipements pour fournir des environnements d'applications logicielles avec un modèle de tarification basé sur l'utilisation des ressources [143].

2. **PaaS :**

Pour la plate-forme en tant que service, de nombreuses plates-formes fournies dans le Cloud sont solution d'application. Ces solutions fournissent généralement des services communs tels que les interfaces utilisateur, le stockage et les bases de données, mais ils sont accessibles uniquement à travers les API de la solution.

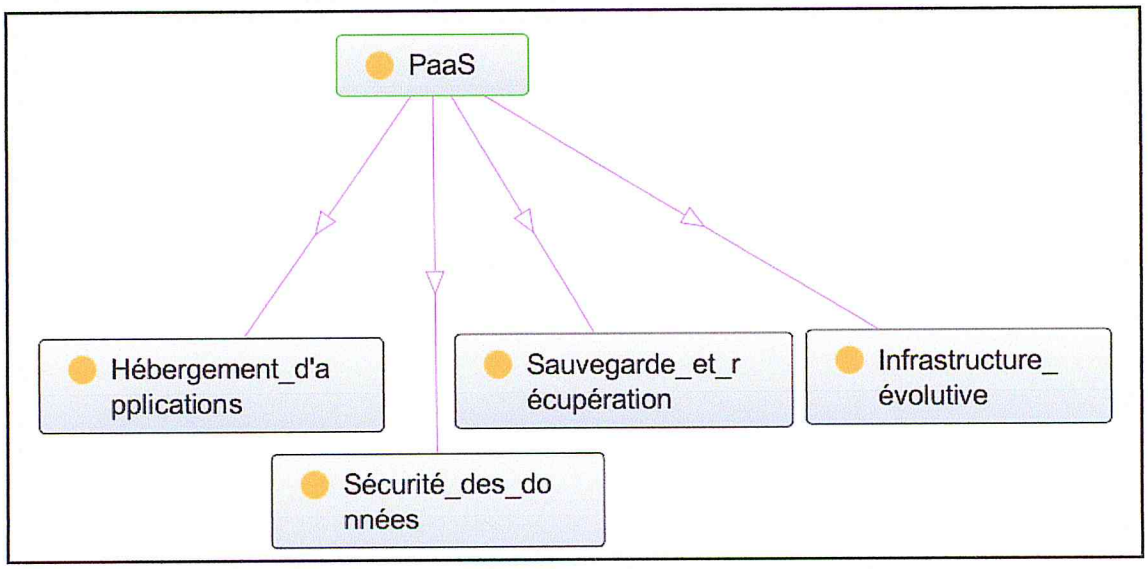


Figure 24 : Ontologie plate-forme en tant que service

La plate-forme en tant que service (PaaS) offre un environnement intégré de haut niveau comme il est représenté dans la (figure 24), afin de créer, tester et déployer des applications personnalisées.

3. **SaaS :**

Pour le logiciel en tant que service, le fournisseur installe, gère et maintient le logiciel. Le fournisseur ne possède pas nécessairement l'infrastructure physique dans laquelle le logiciel fonctionne. Quoi qu'il en soit, le consommateur n'a pas accès à l'infrastructure; Il peut accéder uniquement à la demande.

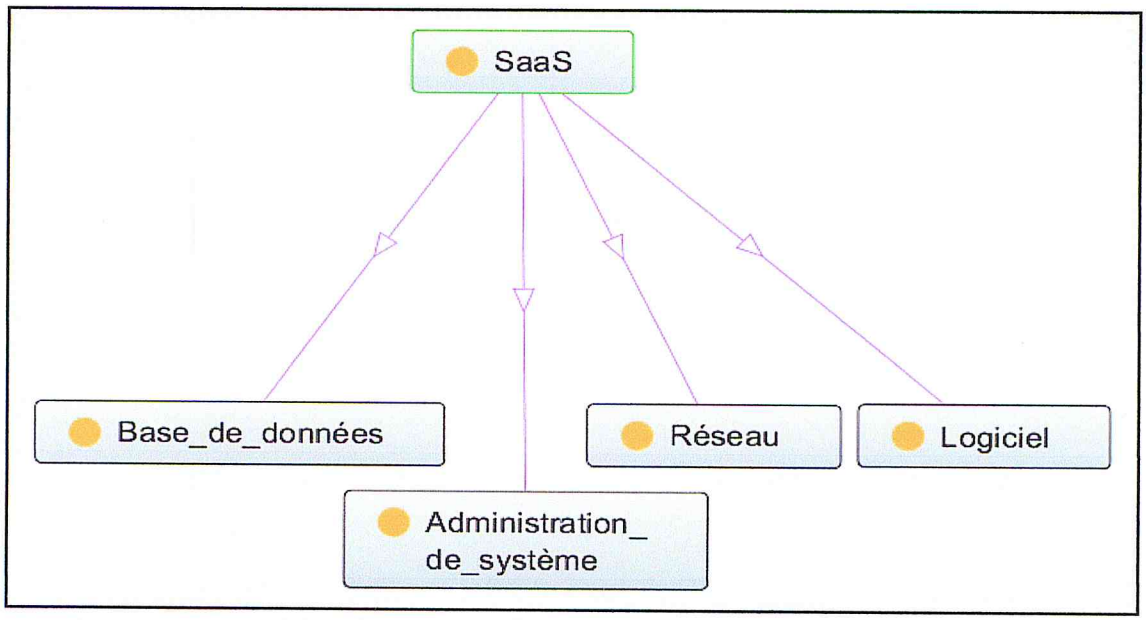


Figure 25 : Ontologie logiciel en tant que service

Comme il est représenté dans la (figure 25), le logiciel en tant que service (SaaS) fournit un logiciel spécialement accessible par les consommateurs via Internet avec un modèle de tarification basé sur l'utilisation.

IV.2.1.3. Liste des règles d'interactions semi-formelles :

La construction des règles d'interaction semi-formelles est une étape très importante dans notre processus, car elle permet de donner une prévision sur la façon dont les interactions devraient être établies, et afin de proposer une solution adaptable à nos prévisions.

Nous prétendons que Sa est le service appartenant au Cloud A et Sb est le service appartenant au Cloud B.

Pre(0) correspond à l'état du service avant son exécution.

Pre(1) correspond à l'état du service avant sa mise en attente.

1)- Un service Sa Pre(0) demandé par B ⟶ si A est libre ⟶ exécuter Sa ⟶ envoyer Sa après son exécution a B.

2)- Un service Sa Pre(1) demandé par B ⟶ si A est occupé ⟶ mise en attente du Sa.

3)- Un service Sb Pre(0) demandée par A ⟶ si B est libre ⟶ exécuter Sb ⟶ envoyer Sb après son exécution a A.

4)- Un service Sb Pre(1) demandé par A ⟶ si B est occupé ⟶ mise en attente du Sb.

IV.2.1.4. Règles d'interaction semi-formelles traduite en langage de prédicats :

- **Les objets :**

Cloud : A , B.
services : Sa , Sb.

- **Les prédicats :**

On (X,Y)	- Vrai si X est le service placé dans le Cloud Y.
Request (X,Y)	- Vrai si X est le service demandé par le Cloud Y.
Check(X)	- Vrai si l'état du Cloud X est vérifié.
Free(X)	- Vrai si X est un Cloud libre (inactif).
Execute(X,Y)	- Vrai si X est le service à exécuter par le Cloud Y.

Wait(X,Y) - Vrai si X est le service mit en attente par le Cloud Y.

Send(X,Y) - Vrais si X est le service a envoyé par le Cloud Y.

Acquire(X,Y) - Vrai si X est le service acquis par le Cloud Y.

- **Etat initial :**

On (Sa, A) et On (Sb, B) sont vrais.

Request (Sa, B) et Request (Sb, A) sont vrais.

- **Etat final :**

On (Sa, B) et On (Sb, A) sont vrais.

- **Les actions:**

1. **Vérification de l'état des services :**

Description : les services doivent être vérifiés après la réception des demandes.

Précondition : On (Sa, A), On (Sb, B), Request (Sa, B), Request (Sb, A) sont vrais.

Effet : Check (A) ,Check (B) ,(Free (A) ou ⌐Free(A)), (Free (B) ou ⌐Free(B)) sont vrais.

2. **Exécution des services demandés :**

Description : les services demandés sont exécutés s'ils sont dans un état libre.

Précondition : On (Sa, A), Free (A), Request (Sa, B), On (Sb, B), Free (B), Request (Sb, A) sont vrais.

Effet : Execute (Sa, A), ⌐Free(A), Output (Sa, A), Execute (Sb, B), ⌐Free(B), Output (Sb, B) sont vrais.

3. **Mise en attente des services demandés :**

Description : les services demandés sont mise en attente s'ils ne sont pas libres.

Précondition : On (Sa, A), ⌐Free(A), Request (Sa, B), On (Sb, B), ⌐Free(B), Request(Sb, A) sont vrais.

Effet : On (Sa, A), Wait (Sa, A), On (Sb, B), Wait (Sb, B) sont vrais.

4. **Envoi des services exécutés :**

Description : les services exécutés sont envoyés aux Cloud intéressés.

Précondition : Execute (Sa, A), On (Sa, A), Output (Sa, A), Execute (Sb, B), On (Sb, B), Output (Sb, B) sont vrais.

Effet : Free (A), Send (Sa, A), Input (Sa, B), ¬Execute (Sa, A), ¬Output (Sa, A), Free(B), Send (Sb, B), Input (Sb, A), ¬Execute (Sb, B), ¬Output (Sb, B) sont vrais.

5. **Acquisition des services envoyés :**

Description : les Clouds envoyant les demandes de services acquièrent les services après leurs envois.

Précondition : Send (Sa, A), Input (Sa, B), ¬Execute (Sa, A), Send (Sb, B), Input (Sb, A), ¬Execute (Sb, B) sont vrais.

Effet : Acquire (Sa, B), On (Sa, B), Acquire (Sb, A), On (Sb, A) sont vrais.

Après la création des règles d'interaction semi-formelles qui représente la première étape qui mène à la résolution de notre problème, nous avons pensé à une solution prenant en considération l'ensemble de règles établit, et nous avons fini par choisir l'algorithme de planification Graphplan comme étant la solution adaptable à la résolution de notre problème.

L'algorithme de planification Graphplan a été déjà utilisé dans la composition de web services en apportant de bon résultat. Notre objectif est d'utiliser cet algorithme pour la technologie du Cloud Computing dans l'espoir que ça va contribuer à résoudre le problème d'interopérabilité sémantique des services Cloud. Mais avant cela, nous devons convertir l'ensemble de règles d'interaction semi-formelles en langage PDDL qui fournit une syntaxe permettant au Graphplan de la prendre en charge.

IV.2.2. Conversion en PDDL des règles d'interaction semi-formelles :

PDDL (Planning Domain Definition Language), est un langage de codage standard pour les tâches de planification "classiques", il est destiné à exprimer le physique d'un domaine, c'est-à-dire les prédicats a utilisé, les actions qui sont possibles, la structure des actions composées et les effets des actions. La plupart des planificateurs exigent une sorte d'annotations à utiliser sur les actions pour atteindre les objectifs ou la réalisation des actions composées, selon les circonstances, ces annotations sont dotées d'une sémantique claire permettant la communication entre les objets [148].

Les composants d'une tâche de planification PDDL sont :

- Objets : Les choses qui nous intéressent.
- Prédicats : Propriétés des objets qui nous intéressent, peuvent être vrai ou faux.
- Etat initial : l'état de démarrage.
- Spécification de l'objectif: résultat que nous voulons obtenir (c'est-à-dire les opérateurs vrai à la fin de planification).
- Actions / Opérateurs : Moyens de changer l'état (passer d'un état à un autre).

Les problèmes de planification représentés dans PDDL sont divisés en deux parties : le domaine et un ou plusieurs problèmes.

- **Le fichier PDDL de problème :** il répertorie le domaine, les objets qui peuvent être utilisés à la place des variables, une description de l'état initial (en utilisant les prédicats répertoriés dans le fichier de domaine) et les critères d'objectif (en utilisant les prédicats).
- **Le fichier PDDL du domaine :** il répertorie les actions disponibles dans le domaine. Il comporte les sections suivantes:
  1. Le nom du domaine.
  2. Une liste de prédicats et les variables de chaque prédicat (un prédicat représente une propriété unique d'un état).
  3. Une liste d'actions, pour chaque action, ses paramètres, conditions préalables et effets qui peuvent être indiqués.

En revenant à notre problème, nous allons utiliser un éditeur PDDL qui affiche l'exécution du fichier problème et domaine (figure 29 et 30) suivants :

IV.2.2.1. Fichier de problème avant planification :

```
1  ;; problem file: cloudcommunication-prob1.pddl
2
3  (define (problem cloudcommunication-prob1)
4    (:domain cloudcommunication)
5    (:objects Sa Sb  A B)
6    (:init (On Sa A)(On Sb B)(Request Sa B)(Request Sb A))
7    (:goal (and (On Sa B)(On Sb A))))
```

Figure 29 : Fichier de problème avant planification

IV.2.3. Planificateur Graphplan :

Supposons que nous avons Sa et Sb deux services correspondant à deux Clouds (Sa correspondant au Cloud A et Sb correspondant au Cloud B), en prenant en compte le problème d'interopérabilité sémantique entre ces deux Clouds via la planification, nous pouvons considérer :

S0 = {On (Sa,A), On (Sb,B)} préposition initiale qui correspond à l'emplacement initial des deux services Sa et Sb.

G = {On (Sa,B), On (Sb,A)} préposition finale qui correspond au but final.

A = {Check(A), Check(B), Execute(Sa,A), Execute(Sb,B), Wait(Sa,A), Wait(Sb,B), Send(Sa,A), Send(Sb,B), Acquire (Sa,B), Acquire (Sb,A)} ensemble d'actions.

Le but de cette planification est de permettre l'interaction des deux services de sorte que le service Sa demandé par le Cloud B se déplace vers le Cloud B après son exécution et le service Sb demandé par le Cloud A se déplace vers le Cloud A après son exécution.

IV.2.3.1. Le graphe de synthèse :

Le graphe de synthèse est représenté dans la (figure 31), les arcs des effets supprimés sont en pointillés alors que ceux des effets ajoutés sont en trais continus et les 'no-op' sont les nœuds noir

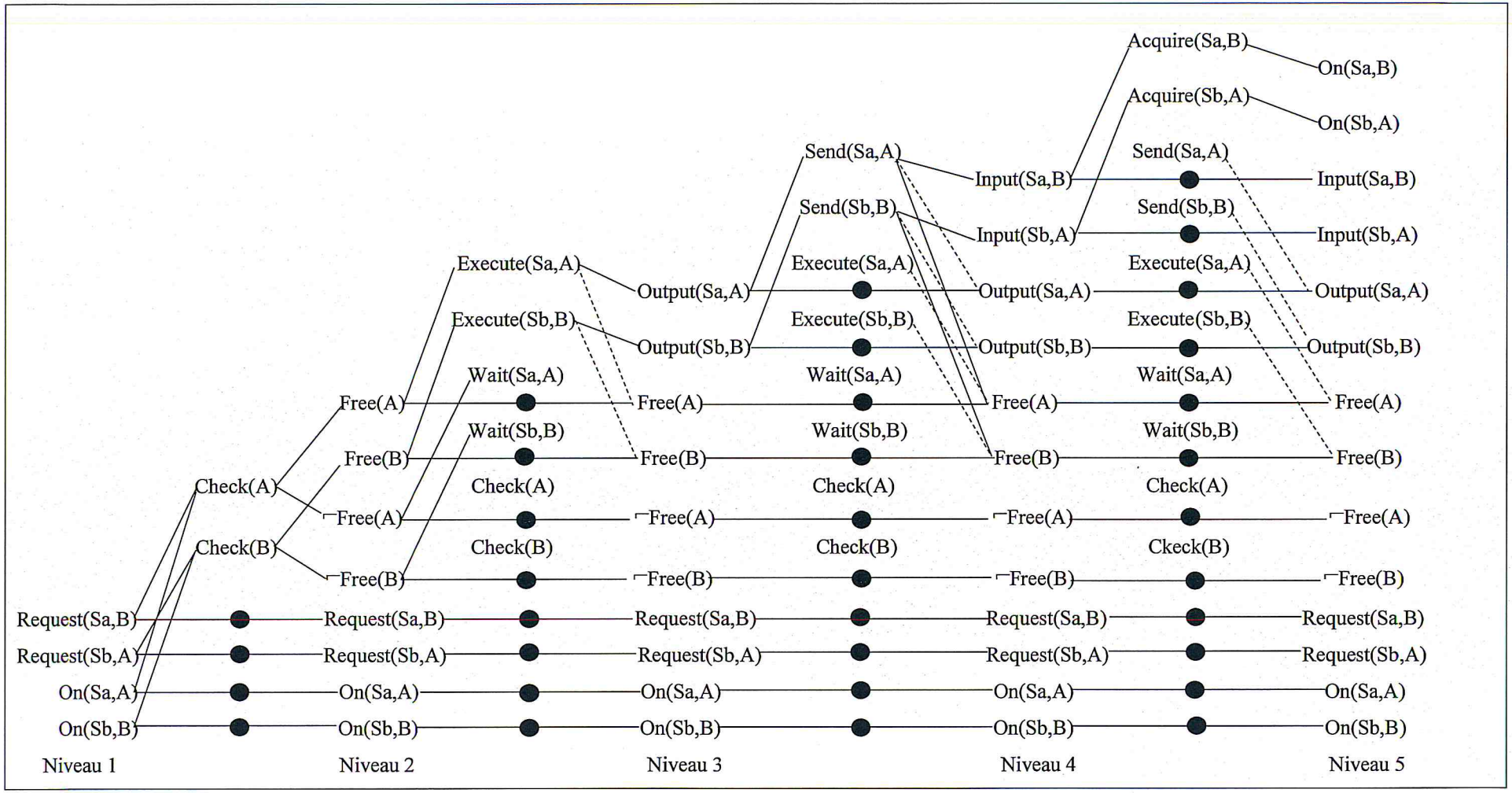


Figure 31 : Graphe de synthèse

IV.2.3.2. Les relations d'exclusions mutuelles entre nœuds :

En formulant le graphe de synthèse, le travail le plus rigoureux à faire est la détection et la propagation des relations d'exclusions mutuelles entre les nœuds d'actions et de propositions.

IV.2.3.2.1. Les actions exclusives :

Deux actions sont 'mutuellement exclusives' dans un même niveau d'actions, si aucun plan valide ne peut les contenir simultanément dans ce même niveau. Plus précisément, deux actions 'a' et 'b' sont marquées exclusives dans un niveau d'actions (i), si on a une des deux conditions suivantes [149] :

- La rivalité : il existe au moins une précondition de 'a' et une précondition de 'b' qui sont mutuellement exclusives dans un niveau de propositions donné.
- L'interférence (non indépendance) : chacune d'entre elles supprime la précondition ou la post-condition de l'autre

Par exemple on a :

1- Execute(Sa,A) et Wait(Sa,A) sont mutuellement exclusive dans le niveau 2 et 3 et 4 car la première a besoin de Free (A) dans sa précondition et la deuxième nécessite ¬Free(A) dans sa précondition, les deux proposition Free(A) et ¬Free(A) sont mutuellement exclusives a un même niveau.

2- Execute(Sb,B) et Wait(Sb,B) sont mutuellement exclusive dans le niveau 2 et 3 et 4 car la première a besoin de Free (B) dans sa précondition et la deuxième nécessite ¬Free(B) dans sa précondition, les deux proposition Free(B) et ¬Free(B) sont mutuellement exclusives a un même niveau.

3- On a aussi Send(Sa,A) et Acquire(Sa,B) sont mutuellement exclusives dans le niveau 4 car la première a besoin de Output(Sa,A) dans sa précondition et la deuxième nécessite Input(Sa,B) dans sa précondition, les deux propositions Output(Sa,A) et Input(Sa,B) sont mutuellement exclusives a un même niveau.

4- On a aussi Send(Sb,B) et Acquire(Sb,A) sont mutuellement exclusives dans le niveau 4 car la première a besoin de Output(Sb,B) dans sa précondition et la deuxième nécessite Input(Sb,A) dans sa précondition, les deux propositions Output(Sb,B) et Input(Sb,A) sont mutuellement exclusives a un même niveau.

Les actions Exécuter et Attente ainsi que envoyer et recevoir provoquent une concurrence pour besoin a un même niveau. Cette concurrence pour besoin se crée quand deux actions de même niveau ont besoin de propositions qui sont mutuellement exclusives.

IV.2.3.2.2. Les propositions exclusives :

Deux propositions sont 'mutuellement exclusives' dans un même niveau, si aucun plan valide ne peut les rendre vraie toutes les deux dans ce même niveau [149]. Dans notre graphe de synthèse on a les propositions mutuellement exclusives suivantes :

Free(A) et ¬Free(A) sont mutuellement exclusives dans les niveaux 2, 3 ,4.

Free(B) et ¬Free(B) sont mutuellement exclusives dans les niveaux 2, 3, 4.

Output (Sa,A) et Input (Sa,B) sont mutuellement exclusives dans le niveau 4.

Output (Sb,B) et Input (Sb,A) sont mutuellement exclusives dans le niveau 4.

Graphplan utilise toutes ces régles relatives aux actions et propositions exclusives pour detécter et enregistrer les relations d'exclusion mutuelles entre les nœuds. La mémorisation sert à propager ces relations à travers les niveau du graphe[149].

IV.2.3.3. Description de l'algorithme :

L'algorithme général de la création du graphe est décrit dans la (figure 32). A partir d'un ensemble (Pi) de conditions initiales, d'une liste des opérateurs permis et d'un ensemble 'G' de buts prédéfinis, la création du graphe se fait par couches, une couche (i) est composée d'un niveau (i) d'actions et d'un niveau (i+1) de propositions [149].

Toutes les conditions initiales sont mises au début dans une table de propositions 'Fact_Table' (ligne 3). Ensuite, tant que tous les buts ne sont pas obtenus de façon non exclusive (ligne4), le planificateur continue à créer les couches avec la procédure 'Create_Graph_Layer' (ligne 5). D'abord, il commence par copier toutes les propositions du dernier niveau propositionnel (i) dans le niveau actuel de

propositions (i+1) qu'il s'apprête à construire (ligne 6). Par la suite, il génère toutes les actions possibles à partir du niveau de propositions (i) ; elles formeront le niveau d'actions (i) (ligne 7). Les arcs de pré-condition sont placés entre les actions obtenues et leurs propositions de pré-conditions dans le niveau de propositions (i) (ligne8). Puis, les post-conditions des actions seront mises dans le niveau (i+1) et les arcs de post-condition ainsi que ceux des effets ajoutés seront également placé (ligne 9-11). Enfin, c'est le tour des actions 'no-op' (ligne 12). Ceci étant fait, les relations d'exclusions mutuelles entre les actions du niveau (i) et celle entre les propositions de niveau (i+1) seront formulées (ligne 13-14).

```
Create_Graph (Operators, Pi, G) {
    Time = 0;
    Load (Pi) in Fact_Table[time] ;
    While (G NOT reached) {
        Create_Graph_Layer (Operators) {
            Copy facts from Fact_Table [time] to Fact_Table[time +1];
            Do_Operators (Fact_Table[time+1], Operators) ;
            Add pre-condition_edges ;
            Put all post-conditions in Fact_Table[time+1] ;
            Add pre-condition_edges ;
            Add delete_edges ;
            Make all no-op;
            Find all mutex actions;
            Find all mutex facts;
        }
        Time = time + 1;
        If (all (g) in G are independent in actuel level) then
            G is reached;
    }
```

Figure 32 : Algorithme de la création du graphe [149]

IV.2.3.4. La recherche du plan valide :

Après la construction du graphe de synthèse et après l'obtention des objectifs totalement indépendants au dernier niveau de propositions, le planificateur fait appel à la procédure de recherche pour chercher un plan valide en débutant à partir des buts et en essayant d'atteindre les conditions initiales du premier niveau propositionnel [149].

Etant donné un ensemble 'G' de buts à un niveau (t), Graphplan essaie de trouver un ensemble d'actions 'no-op' incluses de niveau (t-1) telle que leurs post-conditions soient dans 'G', les préconditions des actions choisies forment l'ensemble 'Sub_Goals' des sous buts de niveau (t-1). En acceptant une action donnée, toutes les propositions constituant sa post-condition seront considérées comme vraies et seront placées dans l'ensemble 'True_Goals'. Pour chaque objectif à examiner de niveau (t), une action de niveau (t-1) l'accomplissant est sélectionnée à condition qu'elle ne soit pas exclusive avec toutes les autres actions retenues dans l'ensemble 'Good_ops' contenant les actions menant aux objectifs déjà examinés. Une fois que tous les buts de niveau (t) ont été examinés et que l'ensemble 'Good_ops' des actions indépendantes qui les accomplissent a été formulé, toutes les préconditions de ces actions (qui figure dans l'ensemble 'Sub_Goals') sont prises comme de nouveaux objectifs à examiner au niveau (t-1) avec la même méthode récursive.

En appliquant la méthodologie de recherche dans notre graphe de synthèse nous obtenons les résultats suivants (figure 33).

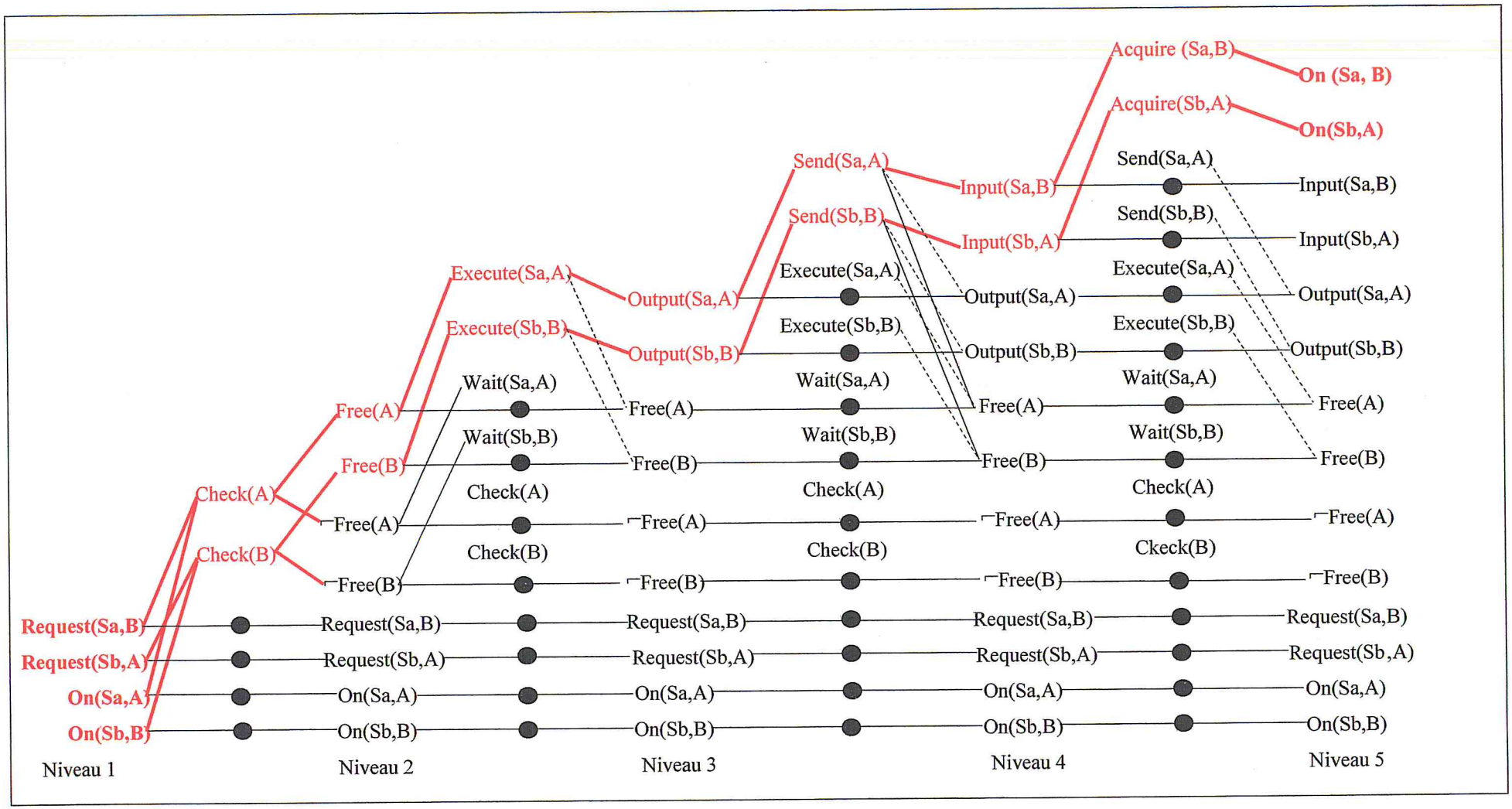


Figure 33 : Procédure de recherche du plan valide

Nous remarquons dans la (figure 33) qu'à partir du chemin obtenu qui est (marqué en rouge) nous sommes parvenus à atteindre les conditions initiales en démarrant des buts finaux et cela en prenant en compte des actions totalement indépendantes, de ce fait nous pouvant dire que nous avons réussi à trouver le plan valide.

IV.2.4. Ensemble de règles d'interaction déduites de la planification :

Après avoir réalisé la planification et recenser les solutions possibles vient l'étape de la formalisation de cette planification, mais avant d'arriver à la formalisation nous devons faire une analyse globale sur notre planification.

Chaque Cloud a une entré et une sortie, les inputs et outputs d'un Cloud représentent les services de type SaaS exécutés après la réception de leur demande. En général, ce que nous voulons faire c'est de planifier les flux entrants et sortants entre deux Clouds en utilisant des règles d'interactions dont leurs formalisations assurent l'interopérabilité des services Cloud.

Etant donné les résultats de la planification établis auparavant en utilisant le Graphplan qui nous a donné une représentation graphique des interactions, nous allons introduire un ensemble de règles d'interactions qui représentent un ensemble d'actions et leurs préconditions décrivant le déroulement des interactions et cela en ajoutant seulement les actions nécessaires à la construction de notre plan, en se basant sur la définition suivante [150]:

L'ensemble des actions ainsi que leurs préconditions dans Graphplan construisant le plan final est une expression de la Forme *(a (*v!) *Pre Add Del)*.

- a est l'ensemble de toutes les actions ;
- v! est la liste des paramètres d'entrés ;
- Pre est la liste des conditions préalables des actions ;
- Add la liste des actions à ajouter ;
- Del est la liste des actions à supprimer.

Suivant cette définition nous obtenant le résultat suivant :

(i) a = {CheckCloud, ExecuteService, Wait, SendService, AcquisitionService}.

(ii) v! = {On (Sa,A), On (Sb,B), Request (Sa, B), Request (Sb, A)}.

(iii)Pre = {On (Sa, A), On (Sb,B), Request (Sa, B), Request (Sb, A), Free (A), Free (B), Execute (Sa, A), Execute (Sb, B), ⌐Execute (Sa, A), ⌐Execute (Sb,B), Output (Sa, A), Output (Sb, B), Send (Sa, A), Input (Sa, B), Send (Sb, B), Input (Sb, A)}.

(iv)Add = {CheckCloud, ExecuteService, SendService, AcquisitionService}

(v) Del = {Wait}.

(vi)Return A = (a (v!)Pre Add Del).

L'ensemble A retourné représente le plan final composé des actions ainsi que leurs préconditions suivantes A = {CheckCloud, ExecuteService, SendService, AcquisitionService, On (Sa, A), On (Sb,B), Request (Sa, B), Request (Sb, A), Free (A), Free (B), Execute (Sa, A), Execute (Sb, B), ⌐Execute (Sa, A), ⌐Execute (Sb, B), Output (Sa, A), Output (Sb, B), Send (Sa, A), Input (Sa, B), Send (Sb, B), Input (Sb, A)}. Les actions dans l'ensemble A représentent l'ensemble d'actions à ajouter afin d'obtenir un plan valide et l'ensemble de préconditions représentent tous les préconditions nécessaires à la réalisation des actions ajoutées, de ce fait l'ensemble A représente l'ensemble des règles d'interactions déduites de la planification.

IV.2.5. Conversion en PDDL des règles d'interactions déduites de la planification:

Convertir l'ensemble de règles d'interactions obtenues en langage PDDL afin de leur donner une représentation formelle en adoptant un format exécutable prouvant leur validité. Les (figures 34 et 35) représentent les fichiers de problème et de domaine après l'application du Graphplan.

IV.2.5.1. Fichier de problème après planification :

```
1  ;; problem file: cloudcommunication-prob1.pddl
2
3  (define (problem cloudcommunication-prob1)
4    (:domain cloudcommunication)
5    (:objects Sa Sb  A B)
6    (:init (On Sa A)(On Sb B)(Request Sa B)(Request Sb A))
7    (:goal (and (On Sa B)(On Sb A))))
```

Figure 34 : Fichier de problème après planification

IV.2.6. Conversion en OWL des règles d'interactions formelles :

Le langage d'ontologie web OWL [151] est conçu pour décrire et représenter un domaine de connaissance spécifique, en définissant des classes de ressources ou objets et leurs relations ; ainsi que de définir des individus et affirmer des propriétés les concernant et de raisonner sur ces classes et individus dans la mesure d'apporter une sémantique formelle du langage OWL.

OWL est un standard basé sur la logique de descriptions [151], il est construit sur RDF et RDFS et utilise la syntaxe RDF/XML.

Le langage OWL permet d'étendre les technologies de base (XML, RDF, RDFS) pour apporter :

- Plus d'interopérabilité (équivalences) ;
- Plus de raisonnements (logique de description) ;
- Plus d'évolution sémantique (intégration d'ontologies).

C'est pour ses avantages que nous l'avons choisi comme étant le langage adaptable à la représentation des données interagi, ainsi que pour la raison d'indisponibilité d'un standard commun pour les services Cloud qui peut provoquer des conflits lors de réception des services interagi en format PDDL.

Afin de réaliser la conversion en OWL des règles d'interactions formelles décrites en langage PDDL dont leur réalisation assure l'interopérabilité sémantique des services Cloud, nous allons construire un méta-modèle qui représente une ontologie des interactions dans un schéma graphique et sous forme d'un programme décrit en langage OWL qui est doté d'une syntaxe claire plus compréhensible.

IV.2.6.1. Méta-modèle des règles d'interactions formelles :

En se basant sur la représentation PDDL des interactions, nous allons construire un méta-modèle (présenté dans la figure 36) qui introduit un ensemble de classes et de propriétés de sorte que les objets et les règles d'interaction dans le programme PDDL vont être représentés dans notre méta modèle en deux classes. Une représente la classe Cloud introduisant l'ensemble d'objets qui représentent les Clouds et l'autre représente la classe de messages introduisant les règles d'interactions. Les propriétés du méta-modèle sont définies dans le (tableau 5) suivant :

Propriété	Domain	Range
HasName	Cloud	String
HasPrototype	Message	String
HasSent	Message	Cloud
HasAcquired	Message	Cloud

Tableau 5 : Propriétés du méta-modèle

Nous prenons en compte une seule interaction des messages du CloudA au CloudB, car l'interaction inverse se fait suivant les mêmes étapes.

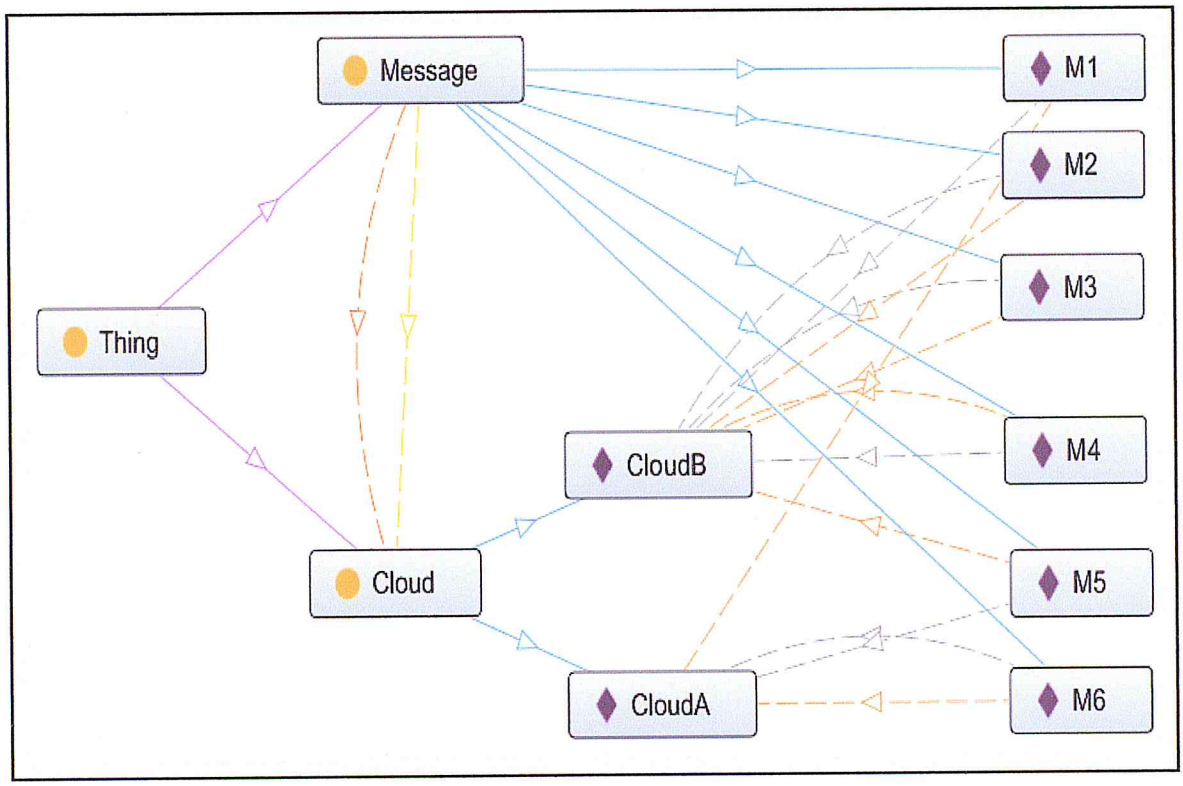


Figure 36 : Méta-modèle des règles d'interactions formelles

IV.2.6.2. Présentation en OWL du méta-modèle :

Cette étape représente la dernière étape dans notre processus, elle représente le méta-modèle établit dans un programme décrit en langage OWL. Il est détaillé dans l'annexe B.

IV.3. Discussions :

Dans ce chapitre, nous avons proposé une solution sémantique pour les problèmes d'interopérabilité des services Cloud. L'idée de base est d'introduire des règles d'interactions formelles permettant la communication entre Clouds, en utilisant un modèle de planification intelligent Graphplan qui permet de construire explicitement une structure compacte appelée Graphique de planification, dans laquelle le plan est une sorte de «flux» présentant les interactions produites.

Avant d'arriver à la planification, nous avons établi un ensemble de règles d'interactions semi-formelles construit sur la base d'un méta-modèle, que nous avons établi afin de rassembler les concepts qui rentrent dans notre champ d'étude. Ces règles d'interaction nous ont donné une idée sur le déroulement des interactions des services entre Clouds. Ensuite, nous avons converti les règles semi-formelles en langage PDDL afin que le planificateur Graphplan puisse les implémentées sous forme de graphe, qui a été décrit par la suite avec un ensemble de règles assurant une représentation simplifiée de la planification, en précisant les actions à ajouter et à supprimer.

Suite à cette étape, nous avons converti la représentation de la planification en langage PDDL qui est doté d'une syntaxe et d'une sémantique formelle adaptable à la machine.

Après la formalisation de la planification, nous avons établi un méta-model représentant les interactions dans une ontologie traduite en langage OWL.

IV.4. Conclusion :

Dans ce chapitre, nous avons présenté une solution pour la résolution du problème d'interopérabilité sémantique entre services Cloud de type SaaS. L'idée de base est d'utiliser un planificateur de l'intelligence artificielle Graphplan pour réaliser la planification des interactions des services Cloud. Nous avons présenté les étapes de la planification et sa formalisation en utilisant le langage PDDL, qui est nécessaire

pour l'utilisation de Graphplan, et le langage OWL introduisant plus d'aspects sémantiques à la planification.

Dans le chapitre suivant, nous allons présenter l'application java dans laquelle nous avons implémenté notre solution.

Chapitre V. *Expérimentation*

V.1. Introduction :

Dans le chapitre précédent de ce mémoire, nous avons proposé une solution basée sur des règles d'interaction assurant l'interopérabilité sémantique des services Cloud (SaaS). Afin d'illustrer les différentes idées et concepts inclus dans la solution proposée, nous allons l'utiliser comme base pour nos simulations.

Pour cela, dans ce chapitre nous allons présenter, dans un premier temps, l'environnement de travail où on va choisir le langage de programmation à utiliser ainsi que l'environnement matériel et logiciel dont nous aurons besoin pour passer par la suite aux étapes de la simulation. Les résultats obtenus à partir de la simulation de notre solution sont présentés dans la dernière section de ce chapitre.

V.2. Langage de programmation java :

Java est un langage de programmation informatique orienté objet et un environnement d'exécution informatique portable créé par James Gosling et Patrick Naughton employés de Sun Microsystems avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld [152]. Java est à la fois un langage de programmation et un environnement d'exécution. Le langage Java a la particularité principale que les logiciels écrits avec ce dernier sont très facilement portables sur plusieurs systèmes d'exploitation tels que : Unix, Microsoft Windows, Mac OS ou Linux avec peu ou pas de modifications... C'est la plate-forme qui garantit la portabilité des applications développées en Java.

Java permet de développer des applications autonomes mais aussi, et surtout, des applications client-serveur. Du Côté client, les applets sont à l'origine de la notoriété du langage, et c'est surtout du côté serveur que Java s'est imposé dans le milieu de l'entreprise grâce aux servlets, le pendant serveur des applets, et plus

récemment les JSP (JavaServer Pages) qui peuvent se substituer à PHP, ASP et ASP.NET.

Les applications Java peuvent être exécutées sur tous les systèmes d'exploitation dans lesquels a été développée une plate-forme Java, dont le nom technique est JRE (Java Runtime Environment - Environnement d'exécution Java). Cette dernière est constituée d'une JVM (Java Virtual Machine - Machine Virtuelle Java), le programme qui interprète le code Java est convertit en code natif, Mais le JRE est surtout constitué d'une bibliothèque standard à partir du quelle tous les programmes en Java doivent être développés [152].

V.3. Environnement de développement NetBeans :

NetBeans est un environnement de développement intégré (EDI) (figure 37), placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refacturent, éditeur graphique d'interfaces et de pages Web) [153].

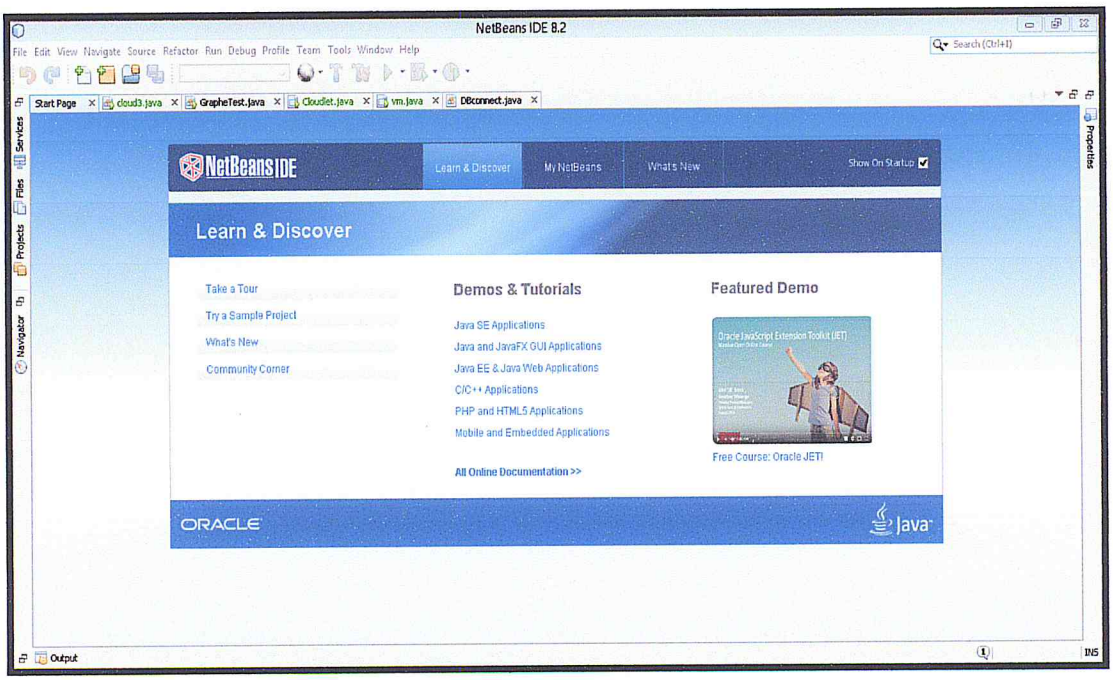


Figure 37 : Environnement de développement NetBeans

V.4. CloudSim :

CloudSim est un nouveau cadre de simulation général, et extensible qui permet la modélisation, la simulation et l'expérimentation de nouvelles infrastructures du Cloud Computing et des services d'application [154].

Dans le domaine du Cloud Computing, les outils de simulations comme CloudSim offre d'importants avantages aux clients et aux fournisseurs. Pour les clients, il permettre de tester leurs services dans un environnement contrôlable avec exempt du coût et de vérifier la performance avant de publier les vrais Clouds, pendant ce temps ils permettent aux fournisseurs de vérifier les types de location en fonction de divers prix et en fonction de la charge [154].

En outre, cela permettra d'optimiser le coût d'accès aux ressources et l'amélioration des bénéfices. Sans ces outils, à la fois des clients et des fournisseurs doit s'appuyer sur des évaluations imprécises ou sur des approches essai-erreur, ces approches peuvent conduire à l'inefficacité de performance des services et la réduction de la génération de revenus. En outre, CloudSim aide les chercheurs et développeurs basés sur l'industrie pour tester la performance d'un service d'application développée dans un environnement convenable et facile à installer. Il existe de nombreux avantages de l'utilisation de CloudSim pour tester les performances de départ, comme: (i) l'efficacité de temps: il prend très moins de temps et d'efforts pour mettre en œuvre des applications de Cloud Computing, (ii) la flexibilité: les développeurs peuvent facilement modéliser et tester les performances de leurs applications et ses services dans des environnements hétérogènes (Microsoft Azure, Amazon EC2) [154].

V.4.1. L'architecture de CloudSim :

La structure logiciel de Cloudsim et ses composants est représenter par une architecture en couche comme il montré par la (figure 38), au niveau le plus bas est le moteur de simulation aux événements discrets SimJava, qui implémente les fonctionnalités de base requises pour les cadres de simulation au niveau supérieur, telles que les files d'attente, le traitement des événements, création de composants du système (services, hôte, Datacenter, Broker, les machines virtuelles), la communication entre les composants et la gestion de l'horloge de simulation[154].

CloudSim supporte la modélisation et la simulation de l'environnement de Datacenter basé sur le Cloud, tel que des interfaces de gestion dédiées aux VMs, la mémoire, le stockage et la bande passante. La couche CloudSim gère l'instanciation et l'exécution des entités de base (VM, hôtes, Datacenter, applications) au cours de la période de simulation. Dans la couche la plus haute de la pile de simulation, on trouve le code de l'utilisateur qui expose la configuration des fonctionnalités liées aux hôtes (ex: nombre de machines, leurs spécifications), les politiques d'ordonnancement de Broker, applications (ex: nombre de tâches et leurs besoins), VM, nombre d'utilisateurs [154].

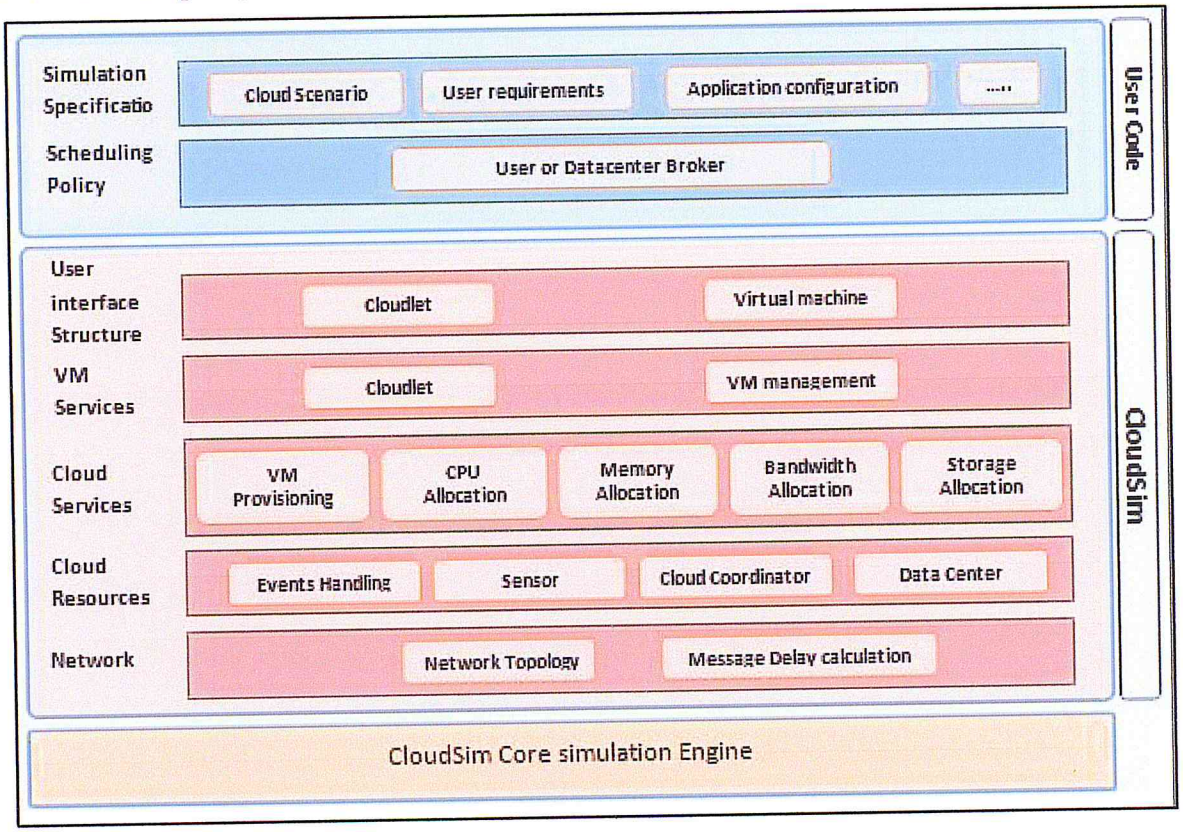


Figure 38 : Les couches de l'architecture CloudSim[155]

V.4.1.1. Cloudlet :

Cette classe modélise les services d'application basés sur le Cloud (tels que la livraison, la gestion des réseaux sociaux, et le déroulement des opérations métier de l'entreprise) qui sont généralement déployer dans des Datacenter. CloudSim orchestre la complexité de la demande en fonction de ses exigences de calcul, représente la complexité d'une application en terme de ces besoin informatique (la taille en nombre

d'instruction, temps de réponse..), Chaque service d'application a un transfert de longueur d'instruction et des données pré-attribué à la fois avant et après la récupération des frais généraux qu'il doit entreprendre au cours de son cycle de vie. Cette classe peut aussi être prorogé pour soutenir la modélisation d'autres mesures de performances et de compositions pour des applications telles que les transactions dans des applications orientées base de données [155].

V.4.1.2. Machine Virtuelle :

Cette classe modélise une instance de machine virtuelle (VM), qui est géré et hébergé pendant son cycle de vie par le composant host Cloud, un host peut simultanément instancier de multiples VMs et assigner des politiques prédéfinies de partage de processeur (espace partagé, temps partagé) [155].

V.4.1.3. Datacenter :

Cette classe modélise l'infrastructure du noyau du service (matériel, logiciel) offert par des fournisseurs de ressources dans un environnement de Cloud Computing. Il encapsule un ensemble de machines de calcul qui peuvent être homogène ou hétérogènes dans leurs configurations de ressources (mémoire, noyau, capacité et stockage). En outre, chaque composant de Datacenter instancie un composant généralisé d'approvisionnement de ressource qui implémente un ensemble de politiques d'allocation de bande passante, de mémoire et des dispositifs de stockage. L'HOST représente un serveur informatique physique dans un Cloud, il exécute des actions liées à la gestion des machines virtuelles et il a une politique définie pour l'approvisionnement de la mémoire et Bw, ainsi que d'une politique de répartition des PE à des machines virtuelles. Un hôte est associé à un DataCentre. Il peut héberger des machines virtuelles [155].

V.4.1.4. DataCentreBroker :

Cette classe modélise le courtier (Broker), qui est responsable de la médiation entre les utilisateurs et les prestataires de services selon les conditions de QoS des utilisateurs, ainsi que le déploiement des tâches de service à travers les Clouds.

Le Broker agissant au nom des utilisateurs identifie les prestataires de services appropriés du Cloud par le service d'information du Cloud CIS (Cloud information

services) en négociant avec eux pour une allocation des ressources qui répond aux besoins de QoS des utilisateurs [155].

V.5. Description de l'application :

V.5.1. Interface principale :

Dès le lancement de notre application, la (figure 39) apparait en premier aux utilisateurs, elle est constituée d'une barre de menu contenant les composants suivants : Fichier, Simulation, Affichage et Aide, chaque composant contient un ensemble d'items. Le menu nous permet d'avoir une idée globale sur le contenu de l'application.

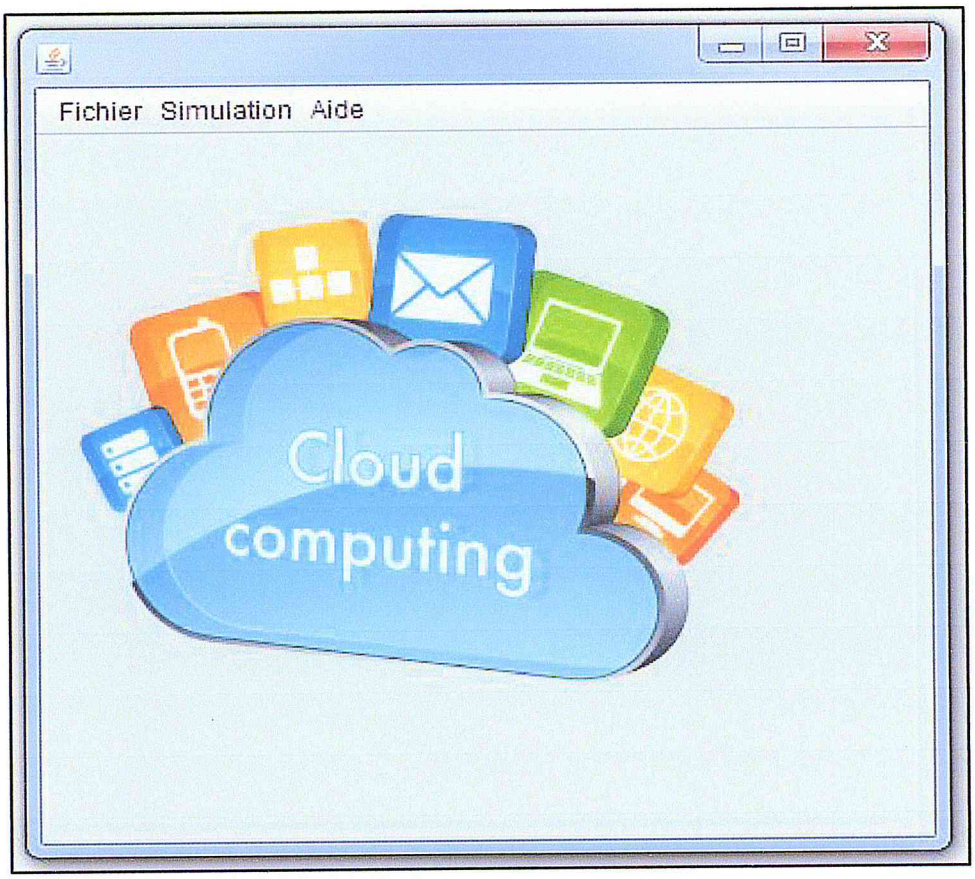


Figure 39 : Interface principale

V.5.2. Configuration des paramètres du Cloud :

Nous allons dans cette partie explorer les interfaces de configuration des paramètres Cloud, ces paramètre sont les mêmes pour tous les Clouds que nous avons

utilisé dans notre application, donc nous avons illustré les interfaces de configuration des paramètres correspondantes a un seul Cloud.

V.5.2.1. Configuration de DataCenter :

Cette étape consiste à la saisi des paramètres nécessaires à la topologie du réseau (figure 40) c'est-à-dire la configuration des paramètres propres au Datacenter comme : le nombre d'instructions exécutés par seconde, la taille de mémoire, la bande passante, ainsi que la taille du stockage de l'host.

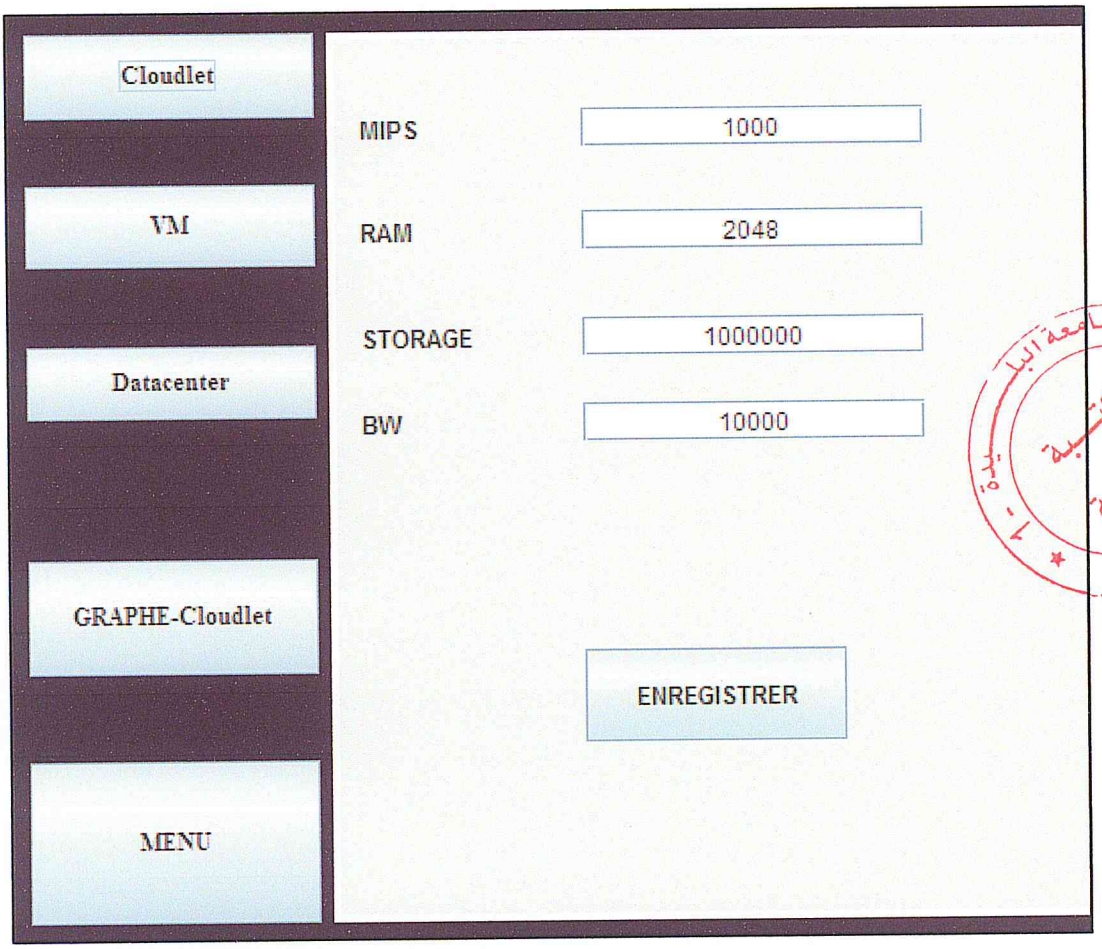


Figure 40 : Configuration du Datacenter

La configuration des Datacenter représente l'affectation de chaque paramètre des valeurs qui correspondent :

- **MIPS :** représente le nombre de millions d'instructions exécutées par seconde, c'est à dire combien de million d'instructions par seconde, MIPS prend la valeur de (MIPS= 500 et plus) selon l'exigence de l'application.

- **RAM :** correspond à la taille de la mémoire qui prend le nombre 2048 (host Memory (MB)).
- **STORAGE** : représente la taille du stockage de l'host qui prend la valeur de 1000000 et plus (host storage (MB)).
- **BW :** correspond à la bande passante, elle prend la valeur de 10000 et plus (BW = 10000 (MB) et plus).

V.5.2.2. Configuration de la Machine Virtuelle :

Cette étape consiste à configurer les paramètres nécessaires propres à des machines virtuelles correspondantes à un Cloud (figure 41), comme la définition de l'identifiant de la machine virtuelle VM, le CPU dans une VM, la spécification de la taille de la RAM du VM, la taille de l'image, le nombre d'instructions exécutées par seconde et la bande passante.

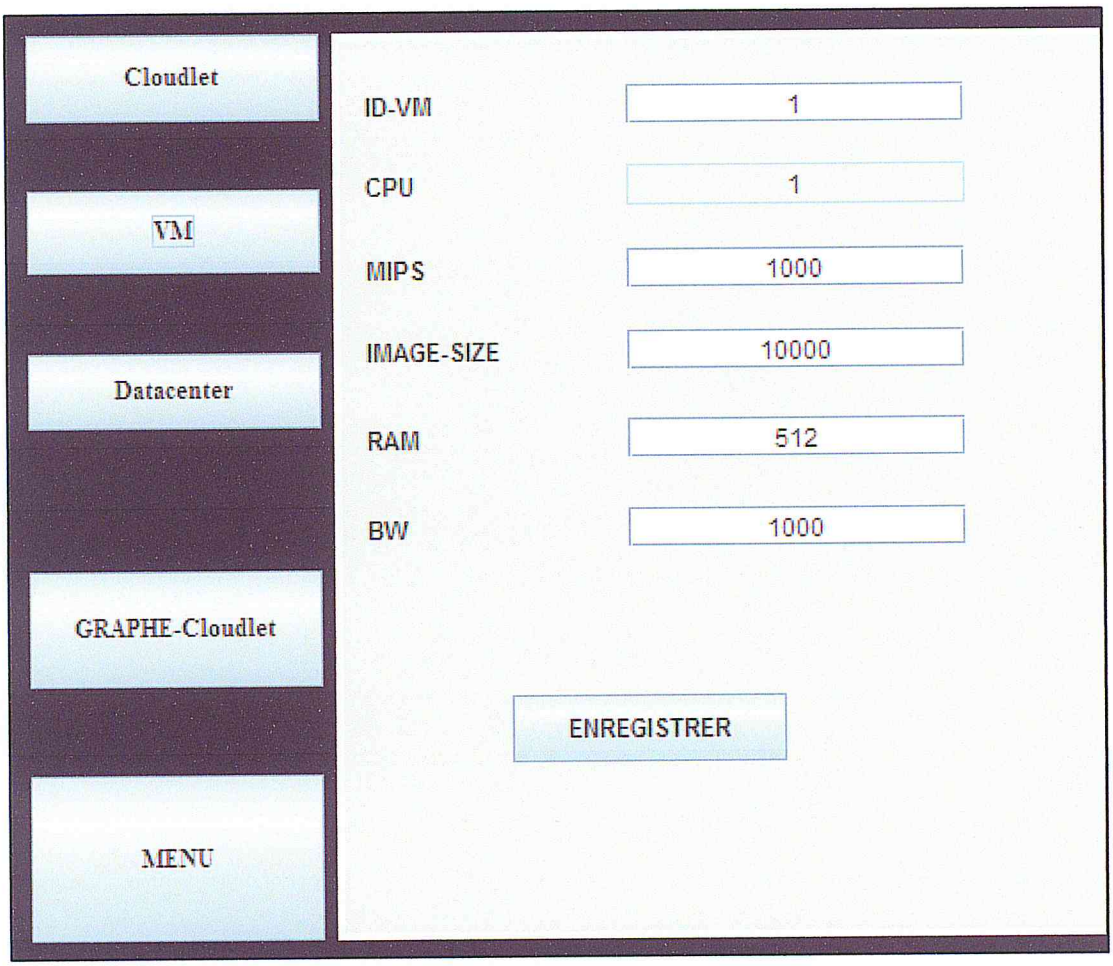


Figure 41 : Configuration de la machine virtuelle

La configuration des machines virtuelles représente l'affectation de chaque paramètre des valeurs qui correspondent :

- **ID-VM :** représente l'identificateur de la VM du Datacenter, il prend une valeur entière positive (**ID-VM** = 0 ou plus).
- **MIPS** : représente le nombre de millions d'instructions exécutées par seconde, c'est à dire combien de million d'instructions par seconde, MIPS prend la valeur de 500 et plus selon l'exigence d'application.
- **IMAGE-SIZE :** représente la taille de l'image qui prend la valeur de 10000 et plus (image size (MB)).
- **RAM :** prend la valeur de 512 et plus (Vm memory (MB)).
- **BW :** La bande passante demandée par VM, elle prend le nombre 1000 et plus (BW = 1000 (MB) et plus).

V.5.2.3. Configuration de Cloudlet :

Dans cette partie, nous allons travailler avec les Cloudlets. Nous allons définir les caractéristiques qui sont attribuées à chacun d'eux (figure 42). Cela signifie que nous pouvons configurer le comportement du Cloudlet : l'identificateur du Cloudlet, la longueur du Cloudlet qui est paramétrée par MIPS et qui influent sur le temps d'exécution de la tâche dans une machine virtuelle, fichier d'entrée et de sortie qui a un impact sur les simulations avec le stockage, le nombre de CPU (PEs) utilisé par Cloudlet qui aura un impact sur la quantité MIPs utilisé pour traiter la tâche.

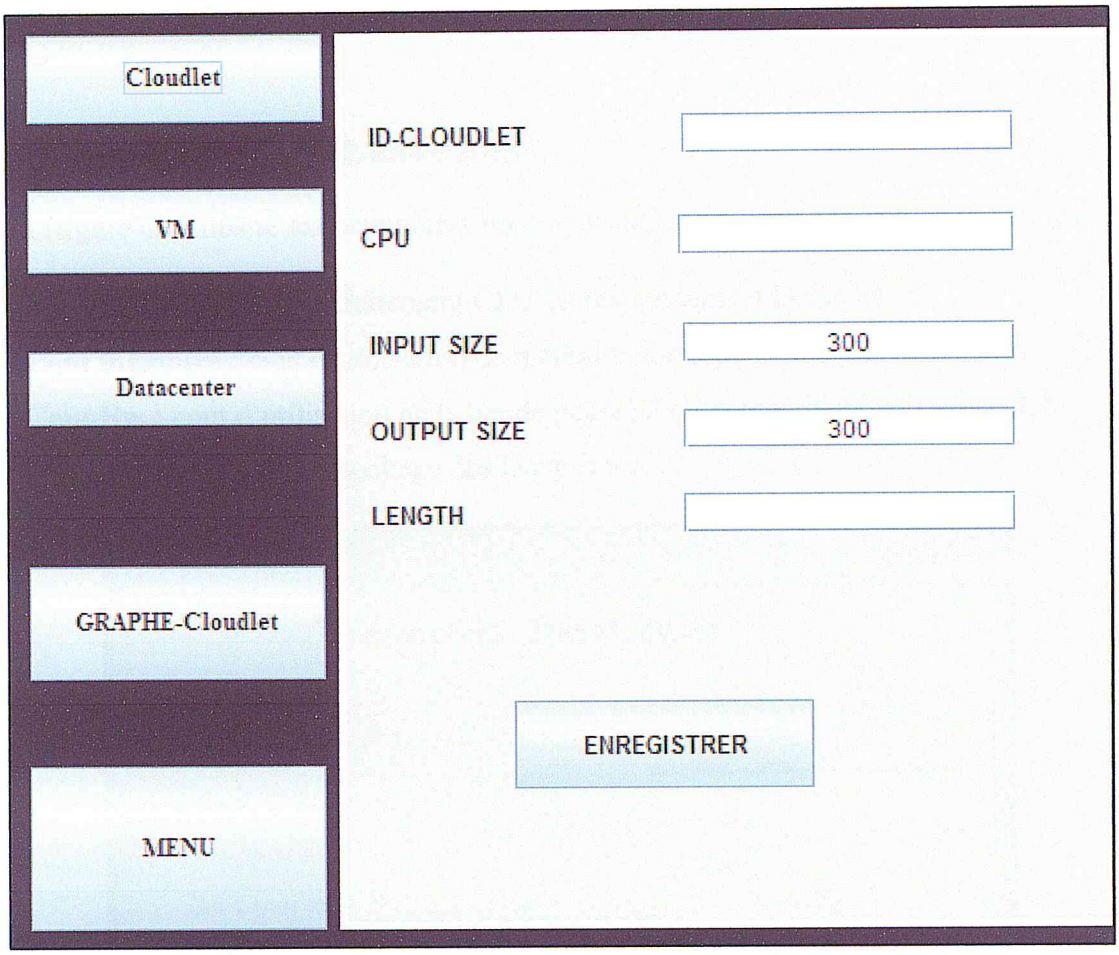


Figure 42 : Configuration du Cloudlet

La (figure 42) montre les paramètres configurés sur l'onglet Cloudlet :

- **ID-CLOUDLET :** représente l'identificateur du Cloudlet, il prend une valeur entière positive.
- **CPU :** le nombre de CPU (PEs) utilisé par Cloudlet, il prend une valeur entière positive.
- **INPUT SIZE** : la taille du fichier d'entrée du Cloudlet avant exécution (la taille du programme + les données en entrée), elle prend un nombre entier positive (exp long fileSize = 300 MB).
- **OUTPUT SIZE :** la taille du fichier de sortie du Cloudlet après exécution (exp long fileSize = 300 MB).
- **LENGTH :** la taille du Cloudlet à exécuter dans le CloudRessource (exp length = 100000 MB).

et qui représente dans notre cas l'ensemble des règles d'interaction à injecter dans nos sockets sous forme de messages interagis.

V.5.5.1. Les sockets :

V.5.5.1.1. Définition :

Un socket est une structure de données abstraite qui est utilisée pour établir un canal de communication permettant l'envoi et la réception d'informations entre des processus qui s'exécutent dans un environnement distribué [156], il est [157] :

- Générique : s'adapte aux différents besoins de communication,
- Indépendant de protocoles et de réseaux particuliers : mais développé à l'origine sous Unix 4BSD, pour Internet.
- N'utilise pas forcément un réseau : par exemple : communication locale (interne à une station).

Le socket est considéré comme une interface de programmation pour les communications entre client et serveur avec un ensemble de primitives qui représentent les identificateurs d'E/S. Nous distinguons la création du socket de son initialisation avec les adresses et les numéros de port [157].

V.5.5.1.2. Principe de fonctionnement :

- **Client :**

Le client peut établir une connexion en demandant la création d'un socket (new Socket()) à destination du serveur pour le port sur lequel le service a été enregistré [158].

- **Serveur :**

le serveur sort de son accept() et récupère un socket de communication avec le client [158].

Le client et le serveur peuvent utiliser des InputSteam et OutputStream pour échanger les données.

Exemple :

Un serveur (programme) s'exécute sur un ordinateur spécifique et possède un socket qui est lié à un port spécifique. Le serveur reçoit la demande de connexion envoyée de la part du client voir la (figure 43a). Si tout se passe bien, le serveur accepte

la connexion voir la (figure 43b). Lors de l'acceptation, le serveur reçoit un nouveau socket lié à un autre port donc il a besoin d'un nouveau socket (par conséquent, un numéro de port différent) afin qu'il puisse accepter les demandes de connexion tout en servant le client connecté [159].

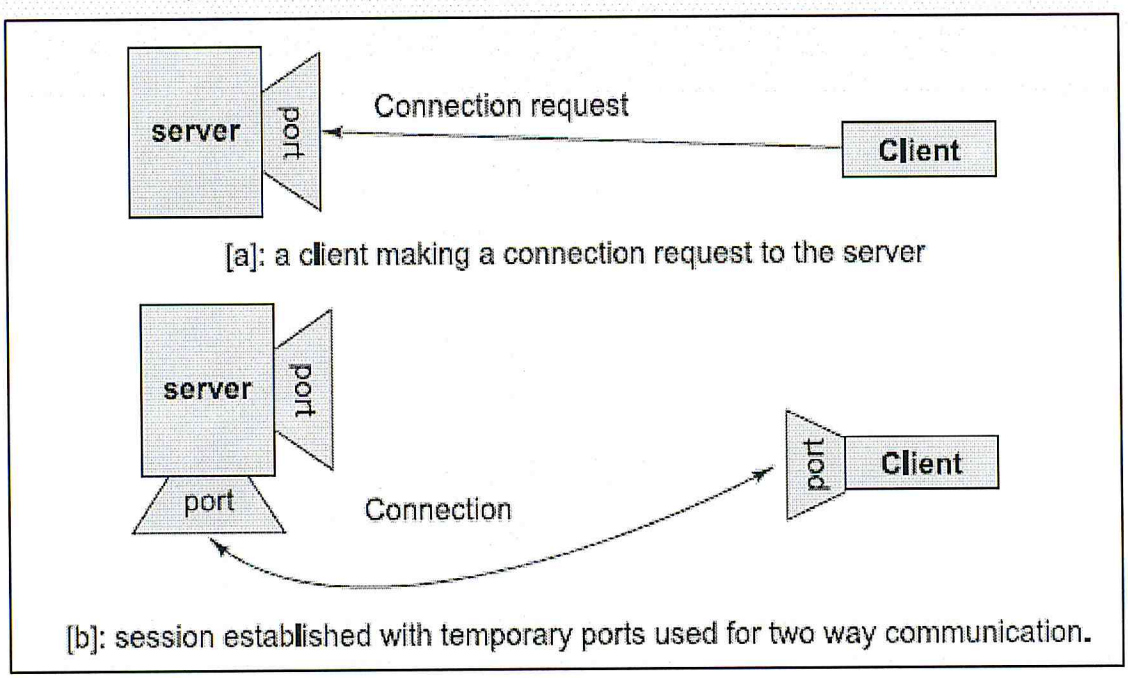


Figure 45 : Établissement d'un chemin de communication bidirectionnel entre un client et un serveur [159]

V.5.5.2. Expérimentation :

Nous allons utiliser les sockets afin de démontrer l'interaction des données entre les deux Clouds, dont l'un jouera le rôle du client et l'autre du serveur.

Dans notre expérimentation nous allons prendre en compte une seule interaction, donc nous allons considérer le Cloud2 comme étant le serveur recevant la demande de charge de la part du client qui représente le Cloud1.

Nous commençons par la première expérience, l'utilisateur remplit les paramètres correspondants au Cloud2, lance le Cloud2, remplit les paramètres correspondants au Cloud1, lance le Cloud1, à partir de là commence l'interopérabilité où nous avons commencé à compter le temps de communication. Après avoir lancer le serveur et le client, la valeur attribuée à (Outputsize) du Cloudlet1 appartenant au

Cloud1 va représenter la valeur de charge demandée de la part du Cloudlet1 au Cloudlet2 appartenant au Cloud2, à son tour le Cloud2 après avoir reçu la demande il modifie la taille du fichier d'entrée (Inputsize) correspondant au Cloudlet2 qui va prendre la même valeur de la charge reçue, ensuite il modifie la taille de son fichier de sortie (Outputsize) qui va prendre la valeur de la taille de son fichier d'entrée (Inputsize), et il envoi par la suite la charge demandée au Cloud1 qui a son tour en recevant la charge modifie la taille de son fichier d'entrée (Inputsize) en fonction de la charge reçue et il finit par lancer la simulation. À partir de là, nous nous arrêtons de compter le temps de communication, car nous avons fini par interagir les charges entre les deux Clouds.

En suivant ces étapes, nous allons réaliser un certain nombre d'expériences dont chacune d'elles représente une interaction, afin d'obtenir un graphe représentant une courbe comme il est présenté dans la (figure 46), qui affiche le développement du temps de communication entre Clouds (Cloud1 et Cloud2), en fonction des valeurs de charges demandées par le Cloud1.

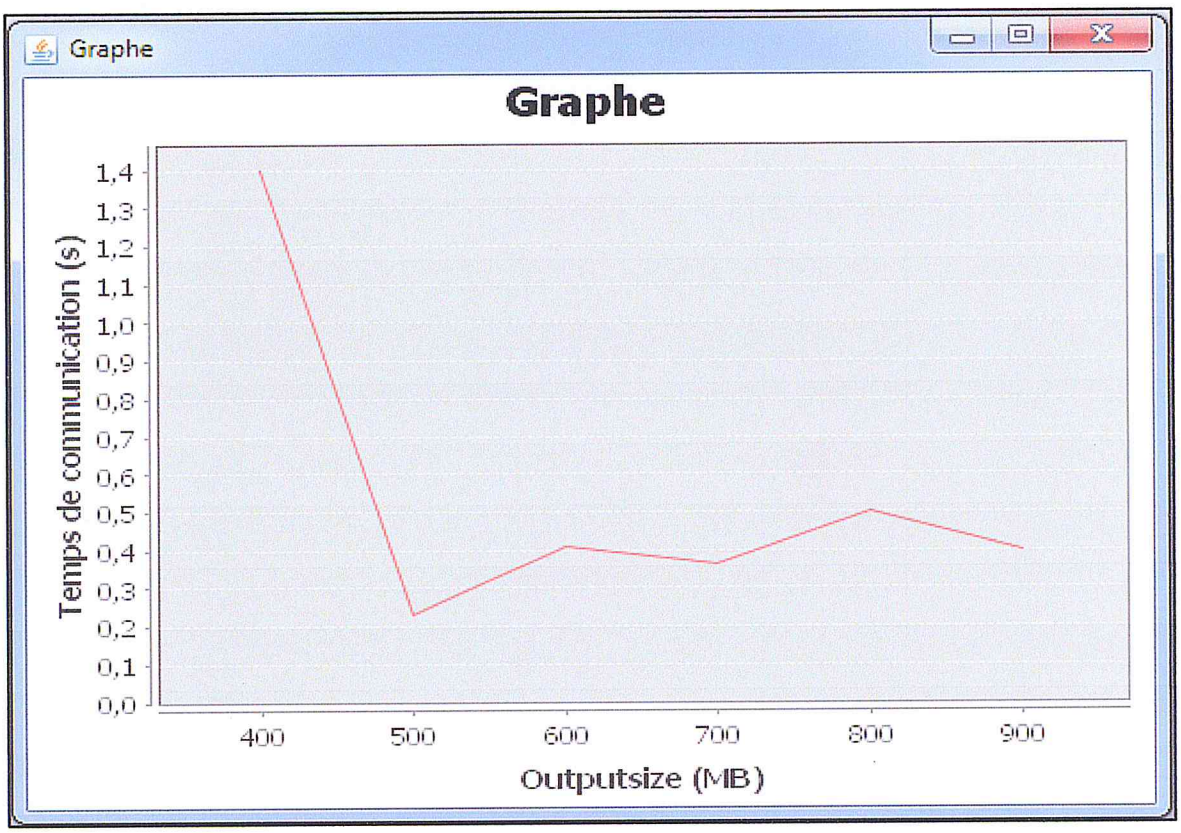


Figure 46 : Graphe de simulation de deux Clouds

D'après les résultats obtenus dans la (figure 46), nous remarquons que les valeurs de temps de communication sont très basses (pas plus de 1.12 ms) comparées aux temps de réponses. De plus, nous remarquons que l'écart de temps entre les expériences établi n'est pas très élevé même en augmentant la valeur de charge demandée, mis à part dans la première itération le temps de communication a pris la valeur de temps la plus élevé et cela du au déclanchement des sockets qui prend un certain moment avant le lancement de la communication.

V.6. Conclusion :

Dans ce chapitre, nous avons illustré une expérimentation de notre solution. Nous avons essayé de mettre en œuvre l'ensemble des idées et concepts qui caractérisent le modèle proposé.

Notre solution a été implémentée en langage JAVA et en utilisant l'outil de simulation du Cloud Computing (CloudSim). Nous avons réalisé deux types de simulations, la première correspond à la simulation d'un seul Cloud dans laquelle nous avons réalisé une expérimentation, qui affiche une courbe mesurant le temps d'exécution en fonction des valeurs attribuées à un paramètre appartenant au Cloudlet. Tandis que la deuxième simulation assure l'interaction des charges entre deux Clouds et cela en utilisant les sockets qui ont pris en considération les règles d'interaction déduites du chapitre précèdent afin de réaliser l'interaction des charges demandées.

Les résultats de nos différentes simulations montrent que le temps de réponse augmente par l'augmentation des charges de Cloud. Mais le temps d'interaction entre les deux Clouds, en appliquant notre solution, reste relativement réduit même en augmentant les valeurs de charges interagis.

Conclusion Générale

Actuellement, l'une des barrières contre l'adoption de systèmes pour les logiciels en tant que service dans l'environnement du Cloud Computing est l'interopérabilité. Après l'étude des travaux connexes, on a constaté que les solutions et les modèles disponibles conçus pour l'interopérabilité sémantique des services Cloud de type logiciels en tant que service ne couvrent pas toutes les exigences de l'interopérabilité sémantique.

Le but de notre travail est de proposer une solution pour l'interopérabilité sémantique des services Cloud de type SaaS. Pour atteindre cet objectif nous avons commencé dans un premier temps par étudier les notions de base du Cloud Computing, après nous avons exploré le problème d'interopérabilité dans les systèmes d'informations en général avant de passer à l'interopérabilité dans le domaine des Clouds. Ce passage était nécessaire pour bien comprendre l'interopérabilité dans les Clouds, car un Cloud représente un système d'information qui déploie ses services à travers internet. Ensuite, nous avons fait une recherche bibliographique sur les travaux qui sont en relation avec notre problématique. Après cette étape, nous avons réalisé une étude comparative des solutions existantes.

L'interopérabilité sémantique des services Cloud de type logiciel en tant que service a été choisie car la plupart des solutions et des modèles existants mettent l'accent sur l'infrastructure en tant que service et plate-forme en tant que service. La recherche de l'interopérabilité sémantique pour les services Cloud de type logiciel en tant que service est encore très immature. Cela, nous a incités durant notre recherche à doubler nos efforts dans le but d'améliorer l'interopérabilité sémantique de ce type de services.

L'idée de notre solution consiste à utiliser la planification de l'intelligence artificielle. Pour cela, nous avons utilisé l'algorithme Graphplan. Le but étant de générer un plan valide introduisant un ensemble de règles d'interactions permettant l'interopérabilité des services (SaaS) entre Clouds.

Pour l'expérimentation de la solution proposée, nous avons utilisé le simulateur Cloudsim. Les tests effectués ont donné de bons résultats surtout en termes de temps d'interopérabilité.

Notre travail présente la première proposition de l'utilisation des Graphplan pour l'interopérabilité sémantique des services Cloud de type SaaS. En effet, pour cette première proposition nous avons choisi le critère de temps d'interopérabilité pour les tests de validation. D'autres critères peuvent être étudiés dans le futur tel que la qualité et le coût d'interopérabilité.

Références

[1] European-Commission. (2004). European interoperability framework for pan-european egovernment services IDA working document, version (Vol. 2).

[2] Cohen, R. (2009). "Examining Cloud Compatibility, Portability and Interoperability". ElasticVapor: Life in the Cloud. http://www.elasticvapor.com/2009/02/examining-cloud-compatibility.html. [Dernière consultation: 01/09/2017].

[3] Moumama Samah, «Vers une plateforme économique avancée pour la gestion des ressource dans les Cloud Computing», Thèse de doctorat, Université d'Oran 1, Algérie, 2015.

[4] Elwessabi Ali Ahmed Yahya, «Une approche basée agent mobile pour le Cloud Computing», mémoire de magister, Université de Batna 2, Algérie, 2014.

[5] Peter Mell and Timothy Grance. The NIST Definition of Cloud Computing. National Institute of Standards and Technology, 2011.

[6] Salvatore D Agostino, Miha Ahronovitz, Joe Armstrong, Rizwan Ahmad, and All. Moving to the Cloud. Technical Report February, Cloud Computing Use Cases Discussion Group, 2011.

[7] G Motta, N Sfondrini, and D Sacco. Cloud Computing: An Architectural and Technological Overview. In Service Sciences IJCSS - 2012 International Joint Conference on Service Sciences, pages 23–27, 2012.

[8] FAREH Mohamed El-kabir, « Une approche basée agents pour l'allocation des ressources dans le Cloud Computing », Mémoire de magister, Université Mohamed Khider – Biskra, Algérie, 2015.

[9] Nicolas Degroodt, « L'élasticité des bases de données sur le Cloud Computing », mémoire de magister, université de Bruxelles, Belgique, 2011.

[10] DataMentors Guides Marketers to Drive Immediate Revenue with Data-as-a-Service. http://www.prweb.com/releases/datamentors/daas/prweb12355856.html. [Dernière consultation : 11/05/2017].

[11] William Gerhardt, Carlos Cordero, Christopher Reberger, Ted Dolan, Mobile Network as a Service A New Solution Architecture for Mobile Network Operators, White Paper, Cisco Internet Business Solutions Group (IBSG), 2013.

[12] IBM Knowledge Center. http://www.ibm.com/support/knowledgecenter/fr/SS4GSP_6.1.3/com.ibm.edt.doc/topics/cloud_connect_amazon_server.html. [Dernière consultation : 11/05/2017].

[13] Jean-Marie Benoist. "La Communication "as a service" – CaaS". http://www.lenouveleconomiste.fr/lesdossiers/la-communication-as-a-service-caas-14963/ . [Dernière consultation: 12/05/2017].

[14] Matt Sarrel. "Cloud Computing -Evaluating Security-as-a-Service-". http://www.cioupdate.com/trends/article.php/3893521/Cloud-Computing---Evaluating-Security-as-a-Service.htm. [Dernière consultation: 13/05/2017].

[15] UNCTAD Technology and Innovation Report. 2011. http://unctad.org/en/Docs/tir2011_en.pdf. [Dernière consultation: 13/05/2017].

[16] Shivaji P Mirashe and N V Kalyankar. Cloud Computing. Journal of computing, 2(3): 78–82, 2010.

[17] Lutz Schubert, Keith Jeffery, and Burkhard Neidecker-Lutz. The Future of Cloud Computing - Opportunities for European Cloud Computing beyond 2010. Technical report, European Commission, 2010.

[18] Sameer Rajan and Apurva Jairath. Cloud Computing: The Fifth Generation of Computing. In 2011 IEEE 3rd International Conference on Communication Software and Networks, volume 15, pages 1–4, 2011.

[19] Keke Gai and Saier Li. Towards Cloud Computing: A Literature Review on Cloud Computing and Its Development Trends. In 2012 Fourth International Conference on Multimedia Information Networking and Security, pages 142–146. IEEE, 2012.

[20] Robert F Roggio, Tetiana Bilyayeva, and James R Comer. Everyday Cloud Computing with SaaS. In The 2012 International Conference on Software Engineering Research and Practice SERP12, 2012.

[21] Sushil Bhardwaj, Leena Jain, and Sandeep Jain. Cloud Computing: A Study Of Infrastructure Aa A Service (IAAS). International Journal of Engineering and Information Technology, 2(1) :60–63, 2010.

[22] Guillaume Plouin, «Cloud Computing Sécurité, gouvernance du SI hybride et panorama du marché», édition Dunod, 2016.

[23] Michael Armbrust, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, and Ariel Rabkin. A view of cloud computing. Communications of the ACM, 53(4): 50–58, 2010.

[24] Michael Armbrust, Anthony D Joseph, Randy H Katz, and David A Patterson. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.

[25] S. Garfinkel, Architects of the information society, Edited by Harold Abelson, 1999.

[26] L. Kleinrock. A vision for the Internet. ST Journal of Research, 2(1): 4-5, November 2005.

[27] I. Foster and C. Kesselman. The grid 2: Blueprint for a new computing infrastructure. Morgan Kaufmann, 2003.

[28] M.A. Vouk. Cloud Computing-issues, research and implementations. Journal of Computing and Information Technology, 16(4): 235-246, 2008.

[29] Amazon EC2. http://aws.amazon.com/en/ec2. [Dernière consultation: 14/05/2017].

[30] Google and IBM announced University Initiative to Adress Internet-Scale Computing Challenges. http://www-03.ibm.com/press/us/en/pressrelease/22414.wss. [Dernière consultation: 14/05/2017].

[31] Google and I.B.M join in 'Cloud Computing' Research. http://www.nytimes.com/2007/10/08/technology/08cloud.html. [Dernière consultation: 15/06/2017].

[32] IBM Introduces Ready-To-Use Cloud Computing. http://www-03.ibm.com/press/us/en/pressrelease/22613.wss. [Dernière consultation: 15/06/2017].

[33] OpenShift. https://Openshift.redhat.com/app/. [Dernière consultation: 16/06/2017].

[34] OpenStack. http://Openstack.org/. [Dernière consultation: 16/06/2017].

[35] Stefan Ried, Holger Kisker, Pascal Matzke, Andrex Bartels and Miroslaw Lisserman. Understanding and quantifying the future of cloud computing. Technical Report, 2011.

[36] Kenneth C. Laudon and Jane Price Laudon. Management Information Systems: Managing the digital firm, 2015.

[37] Celesti, F. Tusa, M. Villari, and A. Puliafito, —How to Enhance Cloud Architectures to Enable Cross-Federation, Proc. IEEE Third Int'l Conf. Cloud Computing (CLOUD), pp. 337-345, 2010.

[38] C. Cachin, R. Haas and M. Vukolic, "Dependable storage in the Intercloud", Research Report RZ, 3783, 2010.

[39] HOGAN MICHAEL, LIU FANG, SOKOL ANNIE ET TONG JIN: NIST cloud computing standards roadmap. NIST Special Publication 35, 2011.

[40] PETCU DANA, Multi-cloud: expectations and current approaches. In Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds, pages 1–6. ACM, 2013.

[41] GOIRI ÍÑIGO, GUITART JORDI ET TORRES JORDI: Economic model of a cloud provider operating in a federated cloud. Information Systems Frontiers, 14(4):827–843, 2012.

[42] PRODAN RADU, WIECZOREK MAREK ET FARD HAMID MOHAMMADI: Double auction-based scheduling of scientific applications in distributed grid and cloud environments. Journal of Grid Computing, 9(4):531–548, 2011.

[43] SOTIRIADIS STELIOS, BESSIS NIK, KUONEN PIERRE ET ANTONOPOULOS NICK: The inter-cloud meta-scheduling (ICMS) framework. In IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), pages 64–73. IEEE, 2013.

[44] Kevin Kelly: A Cloudbook for the Cloud. http://kk.org/thetechnium/a-cloudbook-for/. [Dernière consultation: 18/06/2017].

[45] N. Grozev and R. Buyya. Inter-Cloud architectures and application brokering: Taxonomy and survey. Software Practice and Experience, Softw. Pract. Exper. 2014; 44:369–390.

[46] Adel Nadjaran Toosi, Rodrigo N. Calheiros, and Rajkumar Buyya. Interconnected Cloud Computing Environments: Challenges,Taxonomy, and Survey. ACM Comput. Surv. 47, 1, (May 2014). DOI=10.1145/2593512.

[47] C.-H. Hsu et al. (Eds.): ICA3PP 2010, Part I, LNCS 6081, pp. 13–31, 2010.© Springer-Verlag Berlin Heidelberg 2010.

[48] Himmelman, A.T. "On coalitions and the transformation of power relations: Collaborative betterment and collaborative empowermen." American Journal of Community Psychology 29, n° 2, 2001.

[49] Denise, L. Collaboration vs. C-three (cooperation, coordination, and communication).Innovating 7, n° 3, 1999.

[50] Camarinha-Matos, L, et H. Afsarmanesh. Collaborative Networks: Reference Modeling. Édité par L. Camarinha-Matos et H. Afsarmanesh. Springer-Verlag New York Inc, 2008.

[51] Cassier, J.-L., «Argumentation et conception collaborative de produits industriels », Thèse de doctorat, Université de Grenoble, 2010.

[52] Farouk Belkadi . "Le pilotage des collaborations et l'interopérabilité des systèmes d'information : vers une démarche intégrée". Ecole Centrale de Nantes. Institut de Recherche en Communication et en Cybernétique, Nantes.

[53] Baïna, S., «Interopérabilité dirigée par les modèles : Une Approche Orientée Produit pour l'interopérabilité des systèmes d'entreprise», Thèse de doctorat, Université Henri Poincaré, Nancy I, 2006.

[54] Geraci, A., Katki, F., McMonegal, L., B., M., Lane, J., P., W., Radatz, J., Yee, M., H., P., and Springsteel, F. IEEE Standard Computer Dictionary: Compilation of IEEE Standard Computer Glossaries. IEEE Press, Piscataway,NJ, USA, 1991.

[55] Chen, D. and Doumeingts, G. European initiatives to develop interoperability of enterprise applications - basic concepts, framework and roadmap. Annual Reviews in Control, 2003.

[56] Vernadat, F. Enterprise modelling and integration: principles and applications. Chapman & Hall, 1996.

[57] Wegner, P. Interoperability. ACM Computing Survey, 28(1), 1996.

[58] IEEE. "Standard Computer Dictionary - A Compilation of IEEE Standard Computer Glossaries". Standard Computer Dictionary - A Compilation of IEEE Standard Computer Glossaries. New York, 1990.

[59] Carney, D., D. Fisher, et P. Place. «Topics in Interoperability: System-of-Systems Evolution». Technical Note, Univeristy of Pittsburgh, Software Engineering Institute, 2005.

[60] Chen, D, "Framework for Enterprise Interoperability", White paper, 2010.

[61] Chen, D. & Daclin, N. "Framework for enterprise interoperability". In Bordeaux, France: IFAC 2nd International Workshop on Enterprise Integration, Interoperability and Networking (EI2N'2006).

[62] Obrst, L.J., "Ontologies and Semantic Web for Semantic Interoperability." In McLean, USA: SemanticTechnologies for e-Government Conference, 2004.

[63] C4ISR, Interoperability Working group, "Levels of information systems interoperabilty (LISI)". Architectures Working Group report, US Department of defense, Washington, DC, 1998.

[64] EIF. "European Interoperability Framework." - White Paper, 2004.

[65] Tsuchiya, S. "Improving knowledge creation ability through organizational learning." In Proceedings of International Symposium on the Management of Industrial and Corporate Knowledge ISMICK'93, Compiègne – France, 1993.

[66] Daclin, N., et V. Chapurlat. "Evaluation de l'Interopérabilité Organisationnelle et Managériale des Systèmes Industriels: le projet Carioner." ,2008.

[67] ISO 14528. Automation Systems – Concepts and rules for Enterprise Models, Modelling and architecture. TC 184/SC5/WG1, Geneva, Switzerland: ISO/IEC, 1999.

[68] IEC 62264-1, Enterprise-control system integration, Part 1. Models and terminology. Geneva, Switzerland: ISO/IEC, 2002.

[69] ISO EN DIS 19440. Enterprise integration - Constructs of enterprise modelling, Draft version.TC 184/SC5/WG1, Geneva, Switzerland: ISO/IEC, 2004.

[70] Stroetmann, V., JM Rodrigues, et K. Stroetmann, "Conceptual Framework for eHealth Interoperability." Deliverable 1.1 of the SemanticHEALTH Project, 2006.

[71] ISO-14258-1998, Systèmes d'automatisation industrielle -Concepts et règles pour modèles d'entreprise, International Standard Organization, 1998.

[72] Patrick Hoffmann, «Similarité sémantique inter-ontologies basée sur le contexte», Thèse de doctorat, Université Claude Bernard, Lyon 1,2008.

[73] ISO 14258, Industrial Automation Systems – Concepts and Rules for Enterprise Models, ISO TC184/SC5/WG1, 1999-April-14 version.

[74] DAVID CHEN, "Framework for Enterprise Interoperability", Université Bordeaux 1.2010.

[75] Naudet, Y, "Integrating the Enterprise Interoperability Framework into the Ontology of Interoperability", Research Note, 2007.

[76] A. Sheth and A. Ranabahu, "Semantic Modeling for Cloud Computing, Part I & II," IEEE Internet Computing Magazine, vol. 14, pp. 81-83, 2010.

[77] J. McKendrick. "Does Platform as a Service have interoperability issues?". http://www.zdnet.com/blog/service-oriented/does-platform-as-a-service-have-interoperability-issues/4890. [Dernière consultation : 14/07/2017].

[78] The Economist. "Battle of the Clouds". http://www.economist.com/opinion/displaystory.cfm?story_id=14644393. [Derrière consultation : 14/07/2017].

[79] D. Catteddu and G. Hogben. "Cloud Computing-Benefits, risks and recommendations for information security". http://www.enisa.europa.eu/act/rm/files/deliverables/cloud-computing-risk-assessment/. [Derrière consultation : 15/07/2017].

[80] J. Webster and R. T. Watson, "Analyzing the past to prepare for the future: writing a literature review," MIS Quarterly, vol. 26, pp. 13–23, 2002.

[81] M. B. Becker. "Interoperability Case Study: Cloud Computing", Berkman Center Research Publication No.2012-11, 28 April 2012. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2046987 . [Derrière consultation: 15/07/2017].

[82] M. R. Nelson. "Briefing Paper for the ICCP Technology Foresight Forum – Cloud Computing and Public Policy", Organisation for Economic Co-operation and Development (OECD), document no. DSTI/ICCP(2009)17, 14 October 2009, p.4. http://www.oecd.org/dataoecd/39/47/43933771.pdf. [Derrière consultation: 16/07/2017].

[83] P. Kominers. "Interoperability Case Study: The Internet of Things", Berkman Center Research Publication No.2012-10, 1 April 2012. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2046984 . [Derrière consultation: 16/07/2017].

[84] Open CSA - Open Composite Services Architecture. 2012. http://www.oasis-opencsa.org/. Derrière consultation : [17/07/2017].

[85] Unified Service Description Language. 2012. http://www.w3.org/2005/Incubator/usdl/ . Derrière consultation : [17/07/2017].

[86] Enterprise Mashup Markup Language. 2012. http://www.openmashup.org/omadocs/v1.0/index.html. Derrière consultation : [17/07/2017].

[87] Open Cloud Computing Interface. 2012. http://occi-wg.org/. Derrière consultation: [18/07/2017].

[88] Pahl, C., Zhang, L, Fowley,F., "Interoperability Standards for Cloud Architecture". Dublin City University, Dublin, Ireland.

[89] The Cloud Standards Guide. http://www.cloudwatchhub.eu/cloud-standards-guide. Derrière consultation: [18/07/2017].

[90] Open Virtualization Format. 2012. http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_2.0.0.pdf. Derrière consultation: [18/07/2017].

[91] Nikolaos, L., et all. 2nd IEEE International Conference on Cloud Computing Technology and Science. "Towards a Reference Architecture for Semantically Interoperable Clouds", 2010.

[92] Di Martino, B. Cloud Integration and Standards. "Cloud Portability and Interoperability". Second University of Naples, Italy.

[93] Di Martino, B. "Mapping Design Patterns to Cloud Patterns to support application portability: a preliminary study". Second University of Naples Department of Industrial and Information Engineering Real Casa dell'Annunziata 29, Aversa (CE), Italy.

[94] CLOUDDESIGNPATTERN. "Aws cloud design patterns". http://en.clouddesignpattern.org .Derrière consultation: [20/07/2017].

[95] Microsoft. "Windows Azure Application patterns". https://blogs.msdn.microsoft.com/jmeier/2010/09/11/windows-azure-application-patterns/ . Derrière consultation: [21/07/2017].

[96] Christoph Fehling, Frank Leymann, Ralph Retter, Walter Schupeck, and Peter Arbitter. 2014. "Cloud Computing Patterns". http://cloudcomputingpatterns.org . Derrière consultation: [21/07/2017].

[97] CloudPatterns.org. http://cloudpatterns.org . Derrière consultation: [22/07/2017].

[98] E. PrudHommeaux, A. Seaborne et al., "Sparql query language for rdf," W3C Recommendation, 2008.

[99] Di Martino, B. "Semantic Representation of Cloud Patterns and Services with Automated Reasoning to support Cloud Application Portability". IEEE Transactions on Cloud Computing.

[100] Bergmayr, A. Troya,J. Neubauer,P. Wimmer,M. and Kappel,G. "UML-based Cloud Application Modeling with Libraries, Profiles, and Templates". Business Informatics Group, Vienna University of Technology, Austria.

[101] Bergmayr, A., Bruneliere, H., C´anovas Izquierdo, J.L., Gorro˜nogoitia, J., Kousiouris, G., Kyriazis, D., Langer, P., Menychtas, A., Orue-Echevarria Arrieta, L., Pezuela, C., Wimmer, M.: Migrating Legacy Software to the Cloud with ARTIST. In: CSMR (2013).

[102] Bendoukha Sofiane, «Multi-Agent Approach for Managing Workflows in an Inter-Cloud Environment», Thèse de doctorat, Faculté des mathématiques, informatique et des sciences naturelles, Département de science informatique, Université de Hambourg, 2016.

[103] Wu, Z., Liu, X., Ni, Z., Yuan, D., and Yang, Y. (2013). A marketoriented hierarchical scheduling strategy in cloud workflow systems. The Journal of Supercomputing, 63(1):256–293.

[104] van der Aalst,W. (1998). The application of Petri nets to workflow management. The Journal of Circuits, Systems and Computers, 8(1):21–66.

[105] S. Bechhofer, F. Van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, L. A. Stein et al., "Owl web ontology language reference," vol. 10, pp. 2006–01, 2004.

[106] Ardagna, D., Di Nitto, E., Mohagheghi, P., et al. (2012) Modaclouds: A Model-Driven Approach for the Design and Execution of Applications on Multiple Clouds. ICSE Workshop on Modeling in Software Engineering (MISE), June 2012. IEEE, pp. 50–56.

[107] Menychtas, A., Santzaridou, C., Kousiouris, G., et al. (2013) ARTIST Methodology and Framework: A novel approach for the migration of legacy software on the Cloud. Second Workshop on Management of Resources and Services in Cloud and Sky Computing (MICAS), September.

[108] Bubak, M., Baliś, B., Kitowski, J., et al. (2011) PaaSage: Model-Based Cloud Platform Upperware, Department of Computer Science AGH, Krakow, Poland.

[109] Demchenko, Y., Makkes, M. X., Strijkers, R., and de Laat, C. (2012) Intercloud Architecture for Interoperability and Integration. Fourth International Conference on Cloud Computing Technology and Science (CloudCom), December 2012. IEEE, pp. 666–674.

[110] F. Vernadet et P. "Azena, Prototypage de systèmes d'agents communicants". Journée systèmes Multi-Agents PRC-GRD. Intelligence Artificielle, Nancy, Décembre 1992.

[111] Mazyad Hanaa, «une approche Multi-Agents à architecture P2P pour l'apprentissage collaboratif », Thèse de doctorat, Université du littoral Cote D'opale, 2013.

[112] Younsun, Ahn. Yoonhee, Kim. "Semantic Resource Classification Using Statistical Analysis for Application Characteristics in Intercloud Environment". Dept. of Computer Science Sookmyung Women's University Seoul, Korea.

[113] EuropeanCommission. (2010). Software & Services FP7 Project Portfolio - Internet of Services, Software and Visualisation Call 5. http://cordis.europa.eu/fp7/ict/ssai/docs/2010-3197-call5factsheets-final.pdf .Derrière consultation: [23/07/2017].

[114] Rezaei Reza, «A SEMANTIC INTEROPERABILITY FRAMEWORK FOR SOFTWARE AS A SERVICE SYSTEMS IN CLOUD COMPUTING ENVIRONMENTS», Thèse de doctorat, FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY, UNIVERSITY OF MALAYA,2014.

[115] Mohagheghi, P., Berre, A., Henry, A., Barbier, F., & Sadovykh, A. (2010). REMICS-REuse and Migration of Legacy Applications to Interoperable Cloud Services. Towards a Service-Based Internet, 195-196.

[116] Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente, I.M. Cáceres, J. (2009). The reservoir model and architecture for open federated cloud computing. IBM Journal of Research and Development, 53(4), 4: 1-4: 11.

[117] Garcia-Sanchez, F., Fernandez-Breis, E., Valencia-Garcia, R., Jimenez, E., Gomez, J., Torres-Niño, J., & Martinez-Maqueda, D. (2010). Adding semantics to software-as-a-service and cloud computing. WSEAS Transactions on Computers, 9(2), 154-163.

[118] NEXOFRA. (2010). Deliverable D6.3 The NEXOF Reference Model V3.0. http://www.nexof-ra.eu/sites/default/files/D6.3_v1.0.pdf. Derrière consultation: [23/07/2017].

[119] Cunsolo, V.D., Distefano, S., Puliafito, A., & Scarpa, M. (2009). Volunteer computing and desktop cloud: The cloud@ home paradigm. Paper presented at the Network Computing and Applications, 2009. NCA 2009. Eighth IEEE International Symposium.

[120] Krummenacher, R., Norton, B., Simperl, E., & Pedrinaci, C. (2009). Soa4all: Enabling web-scale service economies. Paper presented at the Semantic Computing, 2009. ICSC'09. IEEE International Conference.

[121] Loutas, Nikolaos, Kamateri, Eleni, Bosi, Filippo, & Tarabanis, Konstantinos. (2011). Cloud computing interoperability: the state of play. Paper presented at the Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference.

[122] Pierre Guisset, Tom Williamson et Keith G. Jeffery. "PaaSage– An €8.4m investment for bridging clouds". https://www.ercim.eu/news/345-paasage .Derrière consultation: [24/07/2017].

[123]The Enterprise Architect. "The Evolution of PaaS in the Entreprise".http://www.theenterprisearchitect.eu/archive/2012/10/17/the-evolution-of-paas-in-the-enterprise. Derrière consultation: [24/07/2017].

[124] The Register. 2013. "Heroku PaaS floats over to Europe". http://www.theregister.co.uk/2013/04/24/heroku_europe/. Derrière consultation: [25/07/2017].

[125] Base One International Corp. "Database Scalability". http://www.boic.com/scalability.htm. Derrière consultation: [26/07/2017].

[126] Keith Jeffrey. 2014. "Keith Jeffrey talks PaaSage at FIA" [video file]. https://www.youtube.com/watch?v=ztMdttXxre0. Derrière consultation: [26/07/2017].

[127] GARCIA-GOMEZ, SERGIO and all. « CHALLENGES FOR THE COMPREHENSIVE MANAGEMENT OF CLOUD SERVICES IN A PAAS FRAMEWORK ». Scalable Computing: Practice and Experience, 2012.

[128] Romano, P. Rodrigues, L. Carvalho, N. Cachopo, J. Cloud-TM: Harnessing the Cloud with Distributed Transactional Memories. INESC-ID/IST.

[129] Aguilera, M. K., Merchant, A., Shah, M., Veitch, A., and Karamanolis, C. Sinfonia: a new paradigm for building scalable distributed systems. In Proc. Symposium on Operating Systems Principles (SOSP) (New York, NY, USA, 2007), ACM, pp. 159–174.

[130] Kotselidis, C., Ansari, M., Jarvis, K., Lujan, M., Kirkham, C., and Watson, I. Distm: A software transactional memory framework for clusters. In Proc. 37th International Conference on Parallel Processing (ICPP) (Sept. 2008), pp. 51–58.

[131] EuropeanCommission. 2011. "Empowering the Service Economy with SLA-aware Infrastructures". http://cordis.europa.eu/project/rcn/87600_en.html. Derrière consultation: [28/07/2017].

[132] Pradhan Karisma, «Semantic Interoperability in Context to Cloud Computing», Thèse de baccalauréat en technologie, NATIONAL INSTITUTE OF SCIENCE & TECHNOLOG PALUR HILLS, BERHAMPUR, ORISSA, INDIA, 2012.

[133] R. Buyya, R. Ranjan, and R. N. Calheiros, "InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services," in Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010), Busan, South Korea, 2010.

[134] R. Buyya, C.-S. Yeo, and S. Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," in Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications (HPCC '08), 2008, pp. 5-13.

[135] Ajith Harshana Ranabahu, Amit P. Sheth, Ashwin Manjunatha, Krishnaprasad Thirunarayan. "Towards Cloud Mobile Hybrid Application Generation using Semantically Enriched Domain Specific Languages". Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis) Center Wright State University, Dayton, Ohio 45435,2012.

[136] Nagarajan, M., Verma, K., Sheth, A.P., Miller, J., Lathem, J.: Semantic Interoperability of Web Services - Challenges and Experiences. IEEE International Conference on Web Services (ICWS) 0, 373{382 (2006).

[137] Couceiro, M., Romano, P., Carvalho, N., and Rodrigues, L. D2STM: Dependable Distributed Software Transactional Memory. In Proc. 15th Pacific Rim International Symposium on Dependable Computing (PRDC) (2009), IEEE Computer Society Press.

[138] Foster, J., (2009). Cloud Computing Standards, Dream vs. Reality. Trend Cloud Security Blog.

[139] Techno-Science.net. "Extreme programming". http://www.techno-science.net/?onglet=glossaire&definition=723 . [Dernière consultation: 14/08/2017].

[140] Piloter.org. "XP eXtreme Programming, les méthodes agiles". https://www.piloter.org/projet/methode/xp.htm . [Dernière consultation: 14/08/2017].

[141] Techno-Science.net. "Ontologie (informatique)". http://www.techno-science.net/?onglet=glossaire&definition=324, [Dernière consultation: 15/08/2017].

[142] Microsoft IECC white paper. Cloud computing use cases, a white paper version 4.0. Cloud Computing Use Case Discussion Group, July 2010.

[143] Ian Foster, Yong Zhao, Ioan Raicu, Shiyong Lu, "Cloud Computing and Grid Computing 360-Degree Compared".

[144] Velte T.Antony. Cloud computer pratical approach. ISBN 978-0-07-162694-1. Hill companies, 2010.

[145] Chalabi, B. «Mise en œuvre d'une solution Cloud Computing avec le modèle MapReduce», Mémoire de magistère, université de M'SILA, 2012.

[146] A. Huth, and J. Cebula. 2012. The Basics of Cloud Computing. U.S CERT.

[147] I. n. Mezga and U. Rauschecker, 2014. The challenge of networked enterprises for cloud computing interoperability. Elsevier, Computers in Industry, Volume 65, Issue 4, pp. 657–674.

[148] Howe, A et all., "PDDL | The Planning Domain Definition Language". AIPS-98 Planning Competition Committee, UCPOP language manual, University of Washington.

[149] Najjar. M.M,.. «Planification par des ordres partiels». 2e cycle, Département des mathématiques et de l'informatique, faculté des sciences, université Sherbrooke, Canada, 2000.

[150]Yang Bo et Qin Zheng. "Semantic Web Service Composition Using Graphplan". School of Electronics & Information Engineering, Xi'an Jiaotong University, china, 2009.

[151] M.K.Smith, C.Welty, D.L.MCGuinness. "OWL Web Ontology Language", http://www.w3.org/TR/2004/REC-owl-guide-20040210/. [Dernière consultation : 17/08/2017].

[152] langage java définition et historique. http://www.comoria.com/3516/Java_%28langage%29 . [Dernière consultation : 20/08/2017].

[153] TheServerSide. "Netbeans". http://www.theserverside.com/definition/NetBeans . [Dernière consultation : 20/08/2017].

[154] LANANI SADOK, «Une approche BPM (Business Process Managment) par composition d'applications dans le Cloud Computing», Mémoire de magister, UNIVERSITE MOHAMED KHIDER, BISKRA.

[155] Jayshri Damodar Pagare et A. Koli Nitin."Design and simulate cloud computing environment using clousimd". Research Scholar Sant Gadge Baba Amravati, University Amravati, India, 2015.

[156] Mohamed Cheriet. "Sockets complément de cours". https://cours.etsmtl.ca/gpa785/Documents/Complement1.pdf . [Dernière consultation : 21/08/2017].

[157] B.Cousin et C.Viho. "Les Sockets". www.irisa.fr/prive/bcousin/Cours/06-socket.fm.pdf . [Dernière consultation : 21/08/2017].

[158] Karima Boudaoud. "Sockets". http://users.polytech.unice.fr/~karima/teaching/courses/I6/cours/module_I6_Sockets.pdf . [Dernière consultation : 22/08/2017].

[159] Socket Programming. http://www.buyya.com/java/Chapter13.pdf . [Dernière consultation : 23/08/2017].

[160] Avrim L. Blum et Merrick L. Furst. Fast Planning Through Planning Graph Analysis. Final version in artificial Intelligence, 90:281-300, 1997.

[161] J. Peer, "Web Service Composition as AI Planning - a Survey," University of St. Gallen, Switzerland, Tech. Rep., 2005.

[162] A. Blum and M. Furst, "Fast Planning Through Planning Graph Analysis,"

in Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 95), 1995, pp. 1636–1642.

[163] Y. Dimopoulos, B. Nebel, and J. Koehler, "Encoding Planning Problems in Nonmonotonic Logic Programs," in In Proceedings of the Fourth European Conference on Planning. Springer-Verlag, 1997, pp. 169–181.

ANNEXES

Annexe A

1. Planificateur Graphplan :

Le planificateur Graphplan proposé par A. Blum et M. Furst [160] génère un graphe de synthèse à partir des conditions initiales en appliquant les actions de base dont il dispose. A partir du graphe, il essaye en appliquant certaines règles de trouver une séquence d'actions menant aux objectifs prédéfinis.

1.1. Le graphe de synthèse :

Le graphe de synthèse construit par Graphplan est un graphe par niveau constitué de deux types de nœuds et de trois types d'arcs. Les niveaux du graphe sont de deux sortes : des niveaux de propositions alternés avec des niveaux d'actions. Les niveaux de propositions contiennent les nœuds de propositions, alors que les niveaux d'actions sont formés par les nœuds d'actions [149].

Le premier niveau du graphe est un niveau de propositions ou chaque nœud y figurant représente une proposition existante parmi les conditions initiales. Un graphe de synthèse est résumé comme suit : des propositions vraies a un temps (t), des actions possiblement applicables a un temps (t), des propositions possiblement vraies a un temps (t+1), des actions possiblement applicables a un temps (t+1), des propositions vraies a un temps (t+2), et ainsi de suite [149].

Les arcs dans le graphe schématisent les relations entre les actions et les propositions. Un nœud d'action dans un niveau d'action (i) est connecté par : (1) les arcs de préconditions à ses nœuds de préconditions situées dans le niveau de propositions (i) , (2) des arcs d'effets ajoutés à ses nœuds de post-conditions qui sont dans le niveau de propositions (i+1), (3) des arcs d'effets supprimés à ses effets supprimés dans le niveau de proposition (i+1) [149].

Dans le graphe, une action peut exister dans un niveau d'action (i) à condition que toutes les propositions de sa précondition existent dans le niveau de propositions (i). Un cas particulier d'actions dites 'no-op' existe dans tous les niveaux du graphe.

Une action 'no-op' est une action neutre. Sa post-condition est exactement identique à sa précondition [149].

Une proposition (p) ne peut exister dans un niveau de propositions (i+1) que si et seulement si elle fait partie de la post-condition d'une action figurant dans le niveau d'actions (i). Cette même proposition (p) pourrait aussi représenter en même temps un effet supprimé d'une autre action de niveau (i) [149].

1.2. Complexité de l'algorithme :

Le temps pris par l'algorithme pour créer un graphe de synthèse est polynomiale en considèrent la longueur de la description du problème et le nombre des étapes.

Théorème : Considérons un problème de planification avec n objets, p propositions dans les conditions initiales, et m STRIPS opérant chacun avec un nombre constant de paramètres formels. Soit l la longueur de la liste d'ajout la plus longue de tous les opérateurs. Ainsi, la taille d'un graphique de planification en t créé par Graphplan et le temps nécessaire pour créer le graphique sont polynomiales n, m, p, l et t [160].

Soit k le plus grand nombre de paramètres formels dans n'importe quel opérateur. Étant donné que les opérateurs ne peuvent pas créer de nouveaux objets, le nombre de propositions différentes qui peuvent être créées par l'instanciation d'un opérateur est $O(ln^k)$. Ainsi, le nombre maximal de nœuds dans n'importe quel niveau de proposition du graphe de planification est $O(p + mln^k)$.

Étant donné que tout opérateur peut être instancié en $O(n^k)$ dans diffèrent cas possible, le nombre maximal de nœuds dans n'importe quel niveau d'action de planification graphique est $O(mn^k)$. De ce fait, la complexité du graphe de planification est polynomiale dans n, m, p, l et t, puisque k est constant [160].

1.3. Avantages d'utilisation de graphplan :

En considérons notre problème d'interopérabilité sémantique via la planification IA, les avantages que procure Graphplan afin de permettre sa réalisation peuvent se résumer comme suit :

• Graphplan réalise un gain énorme en temps en décidant de construire le graphe de synthèse à la première phase avant la recherche de plan. Une fois le graphe créé, il n'a pas besoin d'instancier les actions à partir des opérateurs au cours de la planification, car toutes les actions dont il pourrait avoir besoin ont été formulées au moment de la création du graphe [149].

• Graphplan a de bonnes performances. Le travail majeur dans Graphplan est la création du graphique de planification. La complexité de la création d'un graphique de planification est d'ordre polynômial faible en considérant le nombre d'actions et de propositions, la dépense du temps et de l'espace dans Graphplan est plutôt Plus réduite par rapport à la majorité des planificateurs IA [161].

• Graphplan s'arrêtera automatiquement, quand aucun plan valide n'existe. C'est-à-dire, lorsque deux couches propositionnelles consécutives sont identiques, la construction du Graphique de planification s'arrête [162].

1.4. Limites de Graphplan :

Bien que Graphplan ait beaucoup d'avantages dans la résolution des problèmes de planification, il présente plusieurs défauts en même temps. La performance peut diminuer si les informations contenues dans la spécification d'une tâche de planification sont peut pertinentes [163]. Nous devons réduire la taille du domaine de planification avant la planification, en raison de ce défaut.

Le planificateur utilise son habilité à : (1) déterminer les relations d'exécution mutuelles pour essayer de cerner les contraintes du problème sur lequel il opère et (2) détecter les sous plans parallèles composés d'actions indépendantes. Il est vrai que dans la majorité des cas surtout dans les domaines naturels ces deux atouts facilitent la tache de Graphlan. Mais dans certains cas où il n'y a pas de plans parallèles et ou les relations d'exécutions mutuelles s'avèrent peu efficaces pour la résolution, le planificateur, en absence d'un raisonnement substituant, fonctionne d'une façon ad-hoc, ce qui fait perdre beaucoup de son efficacité [149].

Annexe B

```
<?xml version="1.0"?>
<!DOCTYPE Ontology [
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >]>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
    xml:base="http://www.semanticweb.org/octetplus/ontologies/2017/6/untitled-
ontology-111"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    ontologyIRI="http://www.semanticweb.org/octetplus/ontologies/2017/6/untitled-
ontology-111">
   <Prefix name="" IRI="http://www.w3.org/2002/07/owl#"/>
   <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
   <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
   <Prefix name="xml" IRI="http://www.w3.org/XML/1998/namespace"/>
   <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
   <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
   <Declaration>
      <Class IRI="#Cloud"/>
   </Declaration>
   <Declaration>
      <Class IRI="#Message"/>
   </Declaration>
   <Declaration>
      <ObjectProperty IRI="#HasAcquired"/>
```

```
</Declaration>
<Declaration>
  <ObjectProperty IRI="#HasSent"/>
</Declaration>
<Declaration>
  <DataProperty IRI="#HasName"/>
</Declaration>
<Declaration>
  <DataProperty IRI="#HasPrototype"/>
</Declaration>
<Declaration>
  <NamedIndividual IRI="#CloudA"/>
</Declaration>
<Declaration>
  <NamedIndividual IRI="#CloudB"/>
</Declaration>
<Declaration>
  <NamedIndividual IRI="#M1"/>
</Declaration>
<Declaration>
  <NamedIndividual IRI="#M2"/>
</Declaration>
<Declaration>
  <NamedIndividual IRI="#M3"/>
</Declaration>
<Declaration>
  <NamedIndividual IRI="#M4"/>
</Declaration>
<Declaration>
  <NamedIndividual IRI="#M5"/>
</Declaration>
<Declaration>
```

```
    <NamedIndividual IRI="#M6"/>
</Declaration>
<ClassAssertion>
    <Class IRI="#Cloud"/><NamedIndividual IRI="#CloudA"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="#Cloud"/><NamedIndividual IRI="#CloudB"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="#Message"/><NamedIndividual IRI="#M1"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="#Message"/><NamedIndividual IRI="#M2"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="#Message"/><NamedIndividual IRI="#M3"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="#Message"/><NamedIndividual IRI="#M4"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="#Message"/><NamedIndividual IRI="#M5"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="#Message"/><NamedIndividual IRI="#M6"/>
</ClassAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="#HasAcquired"/><NamedIndividual IRI="#M1"/>
    <NamedIndividual IRI="#CloudB"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="#HasSent"/><NamedIndividual IRI="#M1"/>
```

```
    <NamedIndividual IRI="#CloudA"/>
  </ObjectPropertyAssertion>
  <ObjectPropertyAssertion>
    <ObjectProperty IRI="#HasAcquired"/><NamedIndividual IRI="#M2"/>
    <NamedIndividual IRI="#CloudB"/>
  </ObjectPropertyAssertion>
  <ObjectPropertyAssertion>
    <ObjectProperty IRI="#HasSent"/><NamedIndividual IRI="#M2"/>
    <NamedIndividual IRI="#CloudB"/>
  </ObjectPropertyAssertion>
  <ObjectPropertyAssertion>
    <ObjectProperty IRI="#HasAcquired"/><NamedIndividual IRI="#M3"/>
    <NamedIndividual IRI="#CloudB"/>
  </ObjectPropertyAssertion>
  <ObjectPropertyAssertion>
    <ObjectProperty IRI="#HasSent"/><NamedIndividual IRI="#M3"/>
    <NamedIndividual IRI="#CloudB"/>
  </ObjectPropertyAssertion>
  <ObjectPropertyAssertion>
    <ObjectProperty IRI="#HasAcquired"/><NamedIndividual IRI="#M4"/>
    <NamedIndividual IRI="#CloudB"/>
  </ObjectPropertyAssertion>
  <ObjectPropertyAssertion>
    <ObjectProperty IRI="#HasSent"/><NamedIndividual IRI="#M4"/>
    <NamedIndividual IRI="#CloudB"/>
  </ObjectPropertyAssertion>
  <ObjectPropertyAssertion>
    <ObjectProperty IRI="#HasAcquired"/><NamedIndividual IRI="#M5"/>
    <NamedIndividual IRI="#CloudA"/>
  </ObjectPropertyAssertion>
  <ObjectPropertyAssertion>
    <ObjectProperty IRI="#HasSent"/><NamedIndividual IRI="#M5"/>
```

```
    <NamedIndividual IRI="#CloudB"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="#HasAcquired"/><NamedIndividual IRI="#M6"/>
    <NamedIndividual IRI="#CloudA"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="#HasSent"/><NamedIndividual IRI="#M6"/>
    <NamedIndividual IRI="#CloudA"/>
</ObjectPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#HasName"/><NamedIndividual IRI="#CloudA"/>
    <Literal datatypeIRI="&rdf;PlainLiteral">CloudA</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#HasName"/><NamedIndividual IRI="#CloudB"/>
    <Literal datatypeIRI="&rdf;PlainLiteral">CloudB</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#HasPrototype"/><NamedIndividual IRI="#M1"/>
    <Literal datatypeIRI="&rdf;PlainLiteral">Request</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#HasPrototype"/><NamedIndividual IRI="#M2"/>
    <Literal datatypeIRI="&rdf;PlainLiteral">CheckCloud</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#HasPrototype"/><NamedIndividual IRI="#M3"/>
    <Literal datatypeIRI="&rdf;PlainLiteral">Free</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#HasPrototype"/><NamedIndividual IRI="#M4"/>
```

```
    <Literal datatypeIRI="&rdf;PlainLiteral">ExecuteService</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#HasPrototype"/><NamedIndividual IRI="#M5"/>
    <Literal datatypeIRI="&rdf;PlainLiteral">SendService</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#HasPrototype"/><NamedIndividual IRI="#M6"/>
    <Literal datatypeIRI="&rdf;PlainLiteral">AcquisitionService</Literal>
</DataPropertyAssertion>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#HasAcquired"/><Class IRI="#Message"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#HasSent"/><Class IRI="#Message"/>
</ObjectPropertyDomain>
<ObjectPropertyRange>
    <ObjectProperty IRI="#HasAcquired"/><Class IRI="#Cloud"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#HasSent"/><Class IRI="#Cloud"/>
</ObjectPropertyRange>
<DataPropertyDomain>
    <DataProperty IRI="#HasName"/><Class IRI="#Cloud"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#HasPrototype"/><Class IRI="#Message"/>
</DataPropertyDomain>
<DataPropertyRange>
    <DataProperty IRI="#HasName"/><Datatype abbreviatedIRI="xsd:string"/>
</DataPropertyRange>
<DataPropertyRange>
```

```
    <DataProperty IRI="#HasPrototype"/><Datatype abbreviatedIRI="xsd:string"/>
  </DataPropertyRange>
</Ontology>
<!-- Generated by the OWL API (version 3.4.2) http://owlapi.sourceforge.net -->
```