

UNIVERSITE SAAD DAHLEB DE BLIDA

Faculté des sciences

Département d'informatique



MEMOIRE DE MASTER

En Informatique

Option : Ingénierie du Logiciel

THÈME :

**Solution Broker pour la découverte de
services Cloud Computing de type
Software as a Service**

Réalisé par
AMROUNI Yasmine
AOUINA Sara

Devant le jury
M. DOUGA Yassine (Président)
M. KHEDIRI Abderrezak (Examinateur)
Mme. MANCER Yasmine (Promotrice)

Promotion : 2021/2022

Remerciements

Nous tenons à remercier le bon **Dieu** de nous avoir donné la force et le courage pour accomplir ce modeste travail.

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui nous voulons témoigner toute notre reconnaissance.

Nous tenons à remercier tout particulièrement notre promotrice **Mme. MANCER** pour sa patience, et surtout pour sa confiance, ses remarques et ses conseils, sa disponibilité et sa bienveillance.

Nous remercions sincèrement nos chers parents, nos familles et tous nos amis.

Nous remercions aussi tous les professeurs, intervenantes et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé nos réflexions et ont accepté de répondre à nos questions durant notre recherche.

Nos remerciements vont aussi aux membres de jury qui ont accepté de juger notre travail.

Merci à tous

Dédicace

“

Je dédie ce modeste travail :

À mes parents, Aucun hommage ne pourrait être à la hauteur de l'amour dont ils ne cessent de me combler. Que DIEU leur procure bonne santé et longue vie.

À celui que j'aime beaucoup et qui m'a soutenue tout au long de ce projet : ma soeur Khaoula.

À ma chère binôme Yasmine.

À toute ma famille

Et à tous ceux qui ont contribué de près ou de loin pour que ce Projet soit possible.

Sans oublier tous qui mon enseigné tout au long de mon parcours scolaire.

Merci.

”

Sara

“

Je dédie ce modeste travail :

À la mémoire de mes grands parents maternels Hamama et Mouloud, que DIEU garde leurs âmes dans son vaste paradis.

À la personne la plus chère à mon cœur, ma mère Koulougli Razika pour son amour, sa tendresse, son affection et son soutien inconditionnel.

À mon cher papa Saïd merci infiniment pour tous tes efforts et ta confiance a tes filles qui t'aiment et te respectent.

À mes sœurs Amel, Sabrina, Lamia, Sirine que j'aime beaucoup.

À ma chère binôme Sarah et mes chères amies Maroua, Sihem, Nour, Nourhane, Sabrina, Djomana merci d'être dans ma vie que dieu vous protège.

Sans oublier tous qui mon enseigné tout au long de mon parcours scolaire.

À tous ceux qui me sont chers, à vous tous

Merci.

”

Yasmine

Résumé

Le Cloud Computing est un modèle de fourniture d'infrastructure, de plate-forme et de services logiciels sur le net. Après la grande expansion de l'informatique et Internet, le Cloud Computing est devenu une préoccupation majeure pour les entreprises et les chercheurs dans le domaine d'IT, en raison du grand nombre de services Cloud publiés. Cependant, l'augmentation continue des demandes des utilisateurs, l'hétérogénéité de l'environnement Cloud, le manque de standardisation et les connaissances insuffisantes des utilisateurs sur la technologie Cloud représentent directement ou indirectement les principaux défis liés au Cloud. La découverte et la sélection de services Cloud sont l'un de ces défis qui sont directement touchés par ces problèmes. Les fournisseurs de services Cloud Computing proposent leurs Cloud selon leurs propres règles. Les descriptions de différentes ressources de Cloud diffèrent également pour chaque fournisseur. Il devient ainsi difficile de trouver des services Cloud appropriés pour la satisfaction du besoin d'un utilisateur.

Notre but est de proposer une solution à ces défis pour les services de Cloud Computing public. Cette solution doit garantir une description sémantique des services, une découverte sémantique des services et une sélection basée sur les critères de qualité de services.

Cette solution est destinée aux fournisseurs qui souhaitent mettre à disposition leurs services, mais aussi aux clients qui veulent bénéficier de ces services.

Afin de mettre en œuvre la solution proposée, cette solution a été modélisée conformément à ce qui a été mentionné dans l'étude conceptuelle, une application pour ce modèle est développée en utilisant le langage Java EE.

Mots clés : Cloud Computing, Broker, Description, Découverte, Sélection, Logiciel en tant que service, Qualité de Service, Ontologie, Similarité sémantique.

Abstract

Cloud computing is a model for delivering infrastructure, platform and software services on the net. After the big expansion of IT and the Internet, Cloud Computing has become a major concern for companies and researchers in the field of IT, due to the large number of Cloud services published, the continuous increase in user complexity demands, the heterogeneity of the Cloud environment, Lack of standardization and lack of user knowledge of Cloud technology are directly or indirectly the main problems of many Cloud challenges. The discovery and selection of Cloud services is one of those challenges that are directly affected by these issues. Cloud Computing service providers offer their Cloud services according to their own rules. The descriptions of different Cloud resources also differ for each provider. This makes it difficult to find appropriate Cloud services to meet a user's need.

Our goal is to provide a solution to these challenges for public cloud computing services. This solution must guarantee a semantic description of the services, a semantic discovery of the services and a selection based on the quality of services criteria.

This solution is intended for suppliers who want to make their services available, but also for customers who want to benefit from these services.

In order to implement the proposed solution, this solution has been modeled in accordance with what was mentioned in the conceptual study, an application of this model is developed using Java EE language.

Keywords : Cloud Computing, Broker, Description, Discovery, Selection, Software as a Service, Quality Of Service, Ontology, Semantic similarity.

ملخص

الحوسبة السحابية هي نموذج لتقديم خدمات البنية التحتية والنظام الأساسي والبرمجيات على الشبكة. بعد التوسع الكبير في تكنولوجيا المعلومات والإنترنت ، أصبحت الحوسبة السحابية مصدر قلق كبير للشركات والباحثين في مجال تكنولوجيا المعلومات ، نظرًا للعدد الكبير من الخدمات السحابية المنشورة ، الزيادة المستمرة في متطلبات المستخدم المعقدة، عدم تجانس السحابة ، الافتقار إلى التوحيد القياسي ونقص معرفة المستخدم بالتكنولوجيا السحابية هي المشاكل الرئيسية للعديد من تحديات السحابة بشكل مباشر أو غير مباشر. يعد اكتشاف الخدمات السحابية واختيارها أحد تلك التحديات التي تتأثر بشكل مباشر بهذه المشكلات. يقدم مزودو خدمات الحوسبة السحابية خدماتهم السحابية وفقًا لقواعدهم الخاصة. تختلف مواصفات موارد السحابة المختلفة أيضًا لكل مزود. هذا يجعل من الصعب العثور على الخدمات السحابية المناسبة لتلبية احتياجات المستخدم.

هدفنا هو توفير حل لهذه التحديات لخدمات الحوسبة السحابية العامة يجب أن يضمن هذا الحل وصفًا دلاليًا للخدمات واكتشافًا دلاليًا للخدمات واختيارًا بناءً على معايير جودة الخدمات.

هذا الحل مخصص للموردين الذين يرغبون في إتاحة خدماتهم، ولكن أيضًا للعملاء الذين يرغبون في الاستفادة من هذه الخدمات.

من أجل تنفيذ الحل المقترح ، تم تصميم هذا الحل وفقًا لما تم ذكره في الدراسة المفاهيمية ، تم تطوير تطبيق لهذا النموذج باستخدام لغة EE Java.

كلمات مفتاحية : الحوسبة السحابية، الوسيط، الوصف، الاكتشاف، الاختيار، البرمجيات كخدمة، جودة الخدمة، انطولوجيا، التشابه الدلالي.

Table des Matières

Remerciements	I
Dédicace	II
Résumé	IV
Abstract	V
VI	ملخص
INTRODUCTION GÉNÉRALE	1
I CLOUD COMPUTING	4
I.1 Introduction	4
I.2 Définition du Cloud Computing	4
I.3 Caractéristiques du Cloud Computing	5
I.4 Différents Services du Cloud Computing	6
I.4.1 IaaS « Infrastructure as a Service »	6
I.4.2 PaaS « Platform as a Service »	7
I.4.3 SaaS « Software as a Service »	8
I.4.4 SaaS vs PaaS vs IaaS : Facilité de gestion vs Contrôle complet	10
I.4.5 Autres types de Services Clouds	11
I.5 Composants fonctionnels de Cloud Computing	12
I.6 Taxonomie du marché du Cloud Computing	13
I.7 Accord de niveau de service (SLA)	15
I.7.1 Composants essentiels d'un SLA	15
I.7.2 Caractéristiques du SLA	16
I.8 Fondements technologiques du Cloud Computing	16

I.8.1	Centre de données « Data Center »	17
I.8.2	Virtualisation	17
I.9	Modèles de déploiement du Cloud Computing	18
I.9.1	Cloud public « Public Cloud »	18
I.9.2	Cloud privé « Private Cloud »	19
I.9.3	Cloud Communautaire « Community Cloud »	19
I.9.4	Cloud hybride « Hybrid Cloud »	19
I.10	Avantages du Cloud Computing	20
I.10.1	Flexibilité	20
I.10.2	Déploiement mondial en quelques minutes	20
I.10.3	Efficacité	20
I.10.4	Agilité	21
I.11	Limites du Cloud Computing	21
I.11.1	Sécurité	21
I.11.2	Connectivité	21
I.11.3	Problèmes de transfert de données	21
I.11.4	Perte de données	22
I.12	Conclusion	22
II	LA DÉCOUVERTE DES SERVICES DANS LE CLOUD	23
II.1	Introduction	23
II.2	Description de service Cloud	24
II.2.1	Définition	24
II.2.2	Aspects de description de service Cloud	24
II.2.3	Les langages de description de service	26
II.3	Découverte de service Cloud	27
II.3.1	Définition	28
II.3.2	Environnement de découverte	28
II.3.3	Méthodes de découverte	30
II.4	Sélection de service Cloud	30
II.4.1	Définition	31
II.4.2	Prise de décision multicritère MCDM	31
II.5	Composition des services Cloud	34

II.5.1	Définition	34
II.5.2	Catégorisation des mécanismes de composition	34
II.6	Les problèmes de la découverte des services Cloud	35
II.7	Conclusion	36
III	SOLUTIONS EXISTANTES POUR LA DÉCOUVERTE DES SERVICES DANS LE CLOUD	37
III.1	Introduction	37
III.2	Travaux connexes	37
III.2.1	Solution de Rekik et al [44]	37
III.2.2	Solution de Al-Sayed et al [2]	38
III.2.3	Solution de Abbas et al [3]	39
III.2.4	Solution de Kang et al [45]	40
III.2.5	Solution de Kwang [46]	41
III.2.6	Solution de Han et al [47]	42
III.2.7	Solution de Hasan et al [48]	44
III.2.8	Solution de Beheshti et al [49]	45
III.3	Comparaison des différents travaux	45
III.4	Discussion	49
III.5	Conclusion	49
IV	CONCEPTION DE LA SOLUTION	50
IV.1	Introduction	50
IV.2	Démarche de travail	50
IV.3	Spécification des besoins	51
IV.4	Solution Proposée	53
IV.4.1	Ontologie Cloud :	56
IV.4.2	Dictionnaire Cloud	58
IV.4.3	BDR	59
IV.4.4	Accès aux données OBDA basé sur Ontop	59
IV.4.5	Fonctionnement du SMA	60
IV.4.6	Simulation sur le calcul de similarité	64
IV.4.7	Sélection par MCDM	70
IV.5	Conclusion	73

V	IMPLÉMENTATION DE LA SOLUTION	74
V.1	Introduction	74
V.2	Ressources matérielles	74
V.3	Outils et environnement de développement	74
V.4	Présentation de l'application	80
V.4.1	Espace Fournisseur	82
V.4.2	Espace Utilisateur	92
V.5	Évaluation de la solution par rapport à l'état de l'art	97
V.6	Conclusion	98
	CONCLUSION ET PERSPECTIVES	99

BIBLIOGRAPHIE

WEBOGRAPHIE

ANNEXES

A DÉTAILS DES TRAVAUX CONNEXES

B ALGORITHMES

C CALCUL DE LA SIMILARITÉ

D NOTIONS SUR LES ONTOLOGIES

Table des Figures

FigureI.1	Les services de Cloud Computing [12].	11
FigureI.2	Taxonomie du marché du Cloud Computing [16].	14
FigureIV.1	Diagramme de cas d'utilisation "Cloud Broker"	52
FigureIV.2	Architecture de la solution de découverte de service Cloud proposée	54
FigureIV.3	Les concepts des fonctionnalités des services selon notre ontologie proposée	56
FigureIV.4	La catégorie des DPs selon l'ontologie proposée	57
FigureIV.5	Le processus de Traitement de Texte	61
FigureIV.6	Le processus de mise à jour des mots clés associés	63
FigureIV.7	Domaine de Design	65
FigureIV.8	Domaine de Accounting and Financial	65
FigureIV.9	Exemple de keywords de concept Tax_Accounting	70
FigureV.1	Logo de CloubBroker	80
FigureV.2	Page de présentation du site	80
FigureV.3	Page de présentation du site (suite)	81
FigureV.4	Type d'utilisateur	81
FigureV.5	Interface de connexion	82
FigureV.6	Interface d'inscription	82
FigureV.7	Page d'accueil de fournisseur	83
FigureV.8	Ajouter FF	83
FigureV.9	Confirmation de FF	84
FigureV.10	Exemple d'ontologie Cloud	84
FigureV.11	Ajouter un service en utilisant l'API	85
FigureV.12	Confirmer l'ajout du service	85
FigureV.13	Exemple d'API	86

FigureV.14 Configuration d'API en JSONServer	86
FigureV.15 Data properties	87
FigureV.16 Object properties	87
FigureV.17 Interface de publication d'un service	88
FigureV.18 Interface de publication d'un service (Suite)	88
FigureV.19 Interface de publication d'un service (Suite)	89
FigureV.20 Interface des critères de QoS	89
FigureV.21 Confirmer l'ajout du service	90
FigureV.22 Exemple de FF avec leur Unique et marged Keywords	90
FigureV.23 Exemple de Dictionnaire Cloud	91
FigureV.24 Un extrait de notre BDR	91
FigureV.25 Interface User	92
FigureV.26 Interface "Search for service"	92
FigureV.27 Résultat de découverte	93
FigureV.28 Interface de sélection	93
FigureV.29 Interface Paramètres de service	94
FigureV.30 Interface de SLA	95
FigureV.31 Interface de SLA	95
FigureV.32 Interface de services utilisées	96
FigureV.33 Interface de réputation de service	96
FigureV.34 Confirmation de rating	97
FigureA.1 Les concepts d'ontologie	
FigureA.2 la classe "Acteur".	
FigureA.3 La classe "Service"	
FigureA.4 La classe "PaaS"	
FigureA.5 La classe "Storage_Space"	
FigureA.6 La Classe "TypeVM"	
FigureA.7 La Classe "Price"	
FigureA.8 Les attributs d'ontologie	
FigureA.9 Architecture de cadre de découvert de service Cloud selon la solution d'Al-Sayed et al [2]	
FigureA.10 Exemple de requête de texte utilisateur	

FigureA.11 Interface de services Cloud correspondante récupérée selon la requête d'utilisateur	
FigureA.12 FFs récupérés selon la description de fournisseur	
FigureA.13 Interface ajouter un service HRM	
FigureA.14 Architecture générale des fonctionnalités des services selon l'ontologie CloudFNF	
FigureA.15 Un extrait de l'ontologie CloudFNF se concentrant sur les FFs HRM et SM	
FigureA.16 Structure des données de Dictionnaire Cloud au format JSON	
FigureA.17 Un extrait de BDR se concentrant sur les FFs HRM et SM	
FigureA.18 Un extrait de l'ontologie se concentrant sur les relation Merged et UniqueKeywords sur FFs HRM et SM	
FigureA.19 Architecture de cadre selon la solution d'Abbas et al [2]	
FigureA.20 Les différentes classes de l'ontologie	
FigureA.21 Interface utilisateur pour la découverte des services	
FigureA.22 Résultat de découverte selon la solution de Abbas et al	
FigureA.23 Interface pour la sélection des services Cloud	
FigureA.24 L'interface JADE selon la solution de Abbas et al	
FigureA.25 Interface JADE montre la communication entre les agents de SMA	
FigureA.26 Architecture de cadre selon la solution de Kang et al [45]	
FigureA.27 La procédure de courtage de services cloud basé sur des agents	
FigureA.28 Interface utilisateur et fournisseur	
FigureA.29 Exemple de ancienne approche de recommandation (R1)	
FigureA.30 Exemple de nouvelle approche de recommandation (R2)	
FigureA.31 Architecture de cadre selon la solution de Kwang [46]	
FigureA.32 Une ontologie Windows partielle	
FigureA.33 Architecture de cadre selon la solution de Han et al [47]	
FigureA.34 Interface utilisateur CSDS [47]	
FigureA.35 Architecture de cadre selon la solution de Hasan et al [48]	
FigureA.36 Interface de service Cloud disponible sur le marché [48]	
FigureA.37 Architecture de cadre selon la solution de Beheshti et al [49]	
FigureD.1 Les principales couches du Web Sémantique	

FigureD.2 Syntaxe d'une requete SPARQL

Liste des Tableaux

TableIII.1 Comparaison entre les différentes solutions	48
TableIV.1 Les propriétés de DP selon notre ontologie proposée	57
TableIV.2 Les propriétés de RT selon notre ontologie proposée	58
TableIV.3 Les relations entre les concepts de notre ontologie proposée	58
TableIV.4 Comparaison entre les différentes mesure de similarité et Requête 1	66
TableIV.5 Comparaison entre les différentes mesure de similarité et Requête 2	67
TableIV.6 Comparaison entre les différentes mesure de similarité et Requête 1	68
TableIV.7 Comparaison entre les différentes mesure de similarité et Requête 2	69
TableIV.8 Les services candidats et les valeurs de leurs critères de QoS	71
TableIV.9 La matrice de décision	71
TableIV.10La matrice de décision après la soustraction	72
TableIV.11La matrice de décision après la multiplication (1)	72
TableIV.12La matrice de décision après la multiplication (2)	72
TableIV.13La matrice EWD	73
TableV.1 Évaluation de notre solution	98
TableA.1 Les relations d'ontologie	
TableA.2 Les propriétés de la catégorie RT selon l'ontologie CloudFNF	
TableA.3 Les différentes relations de l'ontologie	
TableA.4 Les différents attributs de l'ontologie	
TableA.5 Les propriétés de l'ontologie	
TableA.6 Les relations de l'ontologie	
TableA.7 Exemple de concept et leurs individus	
TableA.8 Exemple d'individus leur Data type et leur valeur	
TableA.9 Intervalle de chaque propriété	

Liste des Algorithmes

1	Raisonnement des services
2	Algorithme de Matching
3	Algorithme de Découverte
4	Calcul de similarité H_{rel}

Liste des acronymes et abréviations

CRM	<i>Customer Relationship Management</i>
CSD	<i>Cloud Service Description</i>
DP	<i>Deployment Parameters</i>
FF	<i>Functional Functionality</i>
IBM	<i>International Business Machines</i>
IT	<i>Information Technology</i>
JADE	<i>Java Agent DEvelopment Framework</i>
JAWS	<i>Java API for Wordnet Searching</i>
MIT	<i>Massachusetts Institute Of Technology</i>
MPLS	<i>Multiprotocol Label Switching</i>
NFF	<i>Non Functional Functionality</i>
NIST	<i>National Institute Of Standards And Technology</i>
NL	<i>Natural Language</i>
NoSQL	<i>Not Only SQL</i>
OBDA	<i>Ontology Based Data Access</i>

OWL	<i>Web Ontology Language</i>
QoS	<i>Quality Of Service</i>
RDF	<i>Ressource Description Framework</i>
RDFS	<i>Ressource Description Framework Schema</i>
RT	<i>Run Time</i>
SDL	<i>Services Description Language</i>
SGBD	<i>Database Management System</i>
SLA	<i>Service Level Agreement</i>
SLO	<i>Service Level Objectives</i>
SLM	<i>service Level Management</i>
SOA	<i>Service-Oriented Architecture</i>
SPARQL	<i>SPARQL Protocol and RDF Query Language</i>
SQL	<i>Structured Query Language</i>
URI	<i>Uniform Resource Identifier</i>
VDI	<i>Virtual Desktop Infrastructure</i>
VM	<i>Virtual Machine</i>
VPN	<i>Virtual Private Network</i>
WAN	<i>Wide Area Network</i>
WSDL	<i>Web Services Description Language</i>
XML	<i>Extensible Markup Language</i>

Introduction générale

Contexte

Au cours de ces dernières années, avec l'évolution rapide des technologies de l'information, de nombreuses entreprises et organisations ont cherché la meilleure façon pour assurer la mise à l'échelle de leurs systèmes informatiques, réduire les coûts de fonctionnement, fournir une bonne performance à leurs applications tout en optimisant l'utilisation des ressources informatiques. Plusieurs technologies sont apparues au fil des années, tels que, les systèmes distribués, le grid Computing, etc. La technologie que nous allons étudier dans ce mémoire, est le « Cloud Computing ».

Le Cloud Computing est un système de stockage et de calcul de données de la nouvelle génération. Il permet à des utilisateurs de consommer des ressources informatiques en tant que services de différentes natures, avec différents niveaux de contrôle sur les technologies utilisées [1]. Les ressources proposées comprennent des infrastructures, des plateformes de développement et d'exécution ou des applications, ce qui est appelé en anglais Software as a Service (SaaS), Plateforme as a Service (PaaS), Infrastructure as a Service (IaaS). Elles sont généralement hébergées chez un fournisseur de services [1].

Problématique

Le nombre de fournisseurs de services Cloud augmente continuellement, ce qui conduit à augmenter le nombre de services Cloud. Ainsi, les services Cloud publiés souffrent du problème d'hétérogénéité, où les services de la même fonctionnalité publiés à partir d'environnements hétérogènes par différents fournisseurs de Cloud ont des caractéristiques, des descriptions différentes, tel que chaque fournisseur de services Cloud utilise sa propre description de service. De plus, la croissance exponentielle du nombre de services et les connaissances insuffisantes des utilisateurs sur la technologie Cloud représentent directement ou indirectement les prin-

cipaux défis liés au Cloud. La découverte de services Cloud est l'un de ces défis qui sont directement touchés par ces problèmes. Il devient ainsi difficile de trouver des services Cloud appropriés pour la satisfaction du besoin d'un utilisateur [2],[3].

Objectifs

Après avoir pris connaissance des principaux problèmes, nos objectifs sont :

1. Etude de la découverte de services dans le Cloud Computing et dans les services SaaS.
2. Etude comparative des solutions existantes pour la découverte au niveau SaaS.
3. Extraction des problèmes de la découverte des services Cloud de type SaaS.
4. Définir une solution à base de courtier pour la découverte des services Cloud de type SaaS. La solution proposée doit garantir les objectifs suivants :
 - a. Définition d'une solution pour la description des services Cloud de type SaaS. La solution proposée doit couvrir les aspects technique, commercial, opérationnel et sémantique.
 - b. Elaborer les correspondances entre la requête utilisateur et les services SaaS disponibles.
 - c. Sélection des services SaaS basée sur les critères de qualité de services.
5. Validation de la solution proposée par une expérimentation à travers une application Java EE.

Organisation du mémoire

Ce mémoire est subdivisé en cinq principaux chapitres, il commence par une introduction générale, Où nous avons expliqué le concept général de ce mémoire, définissant la problématique et définissant les différents objectifs pour résoudre le problème mentionné.

Le premier chapitre est un chapitre de généralités, il représente une vue globale sur le domaine Cloud Computing, telles que l'historique, la définition du Cloud Computing, les caractéristique essentielles, modèles de déploiement, modèles de service, les avantages et inconvénients, ainsi que l'évolution du Cloud Computing vers l'inter Cloud.

Le deuxième chapitre sera consacré à l'étude de la description, la découverte, la sélection et le composition des services Cloud. Il décrit les dernières technologies concernant les descriptions des services Cloud et les concepts de découverte et la sélection des services. **Le troisième chapitre** est consacré à l'analyse des travaux connexes, dans le but d'extraire les différents critères d'évaluation des solutions, et se termine par une étude comparative de ces travaux.

Le quatrième chapitre est dédiée à la conception de notre solution, qui répond aux problèmes précédents à propos de la description des services Cloud, la découverte des services Cloud et enfin la méthode de sélection utilisée.

Le cinquième chapitre sera consacré à la réalisation de notre solution proposée, dans un premier lieu nous donnons une présentation des outils et l'environnement de développement utilisés. Dans un deuxième lieu, nous présentons une réalisation de notre solution à travers quelques interfaces de notre application Java EE.

Enfin, notre travail s'achève par une conclusion générale résumant les grands points qui ont été abordés ainsi que les perspectives que nous souhaitons accomplir dans le futur.

Chapitre I

Cloud Computing

I.1 Introduction

Actuellement, une nouvelle tendance a fait son apparition dans le monde de technologies de l'information et de la communication, il s'agit du Cloud Computing.

Le Cloud Computing, en français, « informatique dans les nuages », est en train de changer le monde informatique. Il consiste à externaliser des infrastructures informatiques vers des opérateurs spécialisés, au même titre par exemple que les entreprises externalisent la production d'électricité vers des fournisseurs dont c'est le métier principal. Le Cloud a un impact profond sur les utilisateurs et la stratégie informatique de l'entreprise [1].

Au niveau de ce chapitre nous allons présenter le Cloud computing, en spécifiant une définition, voir les types existants, ses caractéristiques, ses modèles de déploiement. Nous décrivons aussi la virtualisation qui est une partie essentielle dans le Cloud Computing, ainsi que les avantages et les inconvénients auxquels il fait face.

I.2 Définition du Cloud Computing

En septembre 2011 NIST [4] a publié dans un rapport sa définition du Cloud Computing qui le considère comme « *un modèle permettant un accès aisé, à la demande et à travers un réseau, à un ensemble partagé de ressources informatiques (par exemple des serveurs, des espaces de stockage, des applications) qui peuvent être rapidement mises en service avec un effort minimum de gestion et d'interaction avec le fournisseur de ce service* ».

L'expression « Cloud Computing » ou « informatique dans les nuages » peut être définie

comme un concept qui consiste à déporter sur des serveurs distants les stockages et les traitements informatiques traditionnellement localisés sur des serveurs locaux ou sur le poste de l'utilisateur [1]. Par conséquent, le traitement informatique n'est plus effectué sur les postes de travail des utilisateurs ou les serveurs internes, mais transfère vers un centre de données « data centers » virtuel maintenu par un fournisseur de services Cloud [1]. Il consiste à proposer des services informatiques sous forme de service à la demande, accessibles de n'importe où, n'importe quand et par n'importe qui. Cette définition est loin d'être simple à comprendre, toute fois l'idée principale à retenir est que le Cloud n'est pas un ensemble de technologies, mais un modèle de fourniture, de gestion et de consommation de services et de ressources informatiques [1].

I.3 Caractéristiques du Cloud Computing

Le modèle Cloud Computing se différencie par les cinq caractéristiques essentielles suivantes :

- **Accès à la demande et en libre-service**

Il s'agit d'un changement majeur dans la manière dont les ressources informatiques sont fournies. Pas besoin de contacter manuellement le fournisseur de services Cloud ou l'équipe informatique de l'organisation pour allouer des ressources ou modifier des configurations spécifiques [5]. Les utilisateurs de services Cloud s'attendent à un accès à la demande et presque instantané aux ressources. Pour répondre à cette attente, le Cloud doit permettre un accès en libre-service « Self-Service » afin que les clients puissent demander, personnaliser, payer et utiliser des services sans l'intervention d'opérateurs humains [6].

- **Accès étendu au réseau**

Quel que soit l'appareil utilisé (PC, Mac, tablette, smartphone, etc.), vous pouvez accéder à des environnements de type Cloud Computing depuis plusieurs emplacements via une connexion réseau standard.

- **Élasticité rapide**

Le Cloud donne l'illusion de ressources informatiques (presque) illimitées et disponibles à la demande. L'élasticité rapide est la capacité à augmenter et à réduire rapidement ces services. Les ressources peuvent être rapidement étendues ou réduites en fonction de la demande. Les utilisateurs peuvent donc affecter des ressources, payer pour la période allouée et, lorsque les ressources ne sont plus nécessaires, ils peuvent simplement les

libérer à nouveau dans le réservoir de ressources ce qui permet d'éviter le coût des ressources informatiques inactives et réagir rapidement pour continuer à faire évoluer la demande [5].

- **Mise en commun des ressources**

La mise en commun ou la mutualisation des ressources est essentiellement la possibilité d'utiliser un réservoir (pool) de ressources pour plusieurs utilisateurs finaux, clients et consommateurs. Les ressources utilisées pour fournir des services dans le Cloud sont mises en œuvre à l'aide d'une infrastructure homogène, partagée entre tous les utilisateurs du service, tout en protégeant la sécurité et les données privées de chaque client. Cela est possible en utilisant des outils de virtualisation qui permettent le partage de serveurs entre plusieurs utilisateurs [7].

- **Service mesuré en permanence**

Les informations de consommation sont également disponibles pour que les consommateurs puissent suivre leur propre consommation, recevoir des alertes basées sur des limites configurables, afficher des rapports, auditer les opérations en cours sur le système et surtout utiliser ces informations pour optimiser leurs utilisations et réduire les coûts. Le Cloud Computing permet au client de payer à l'usage, uniquement ce qui a été consommé selon le type de service (stockage, traitement, bande passante et comptes d'utilisateurs actifs, etc.) et de la durée d'utilisation du service [7].

I.4 Différents Services du Cloud Computing

Le Cloud Computing permet aux entreprises d'utiliser des services informatiques par facturation à la demande et à l'utilisation. Ces services peuvent être montrés sous de nombreuses formes, qui sont :

I.4.1 IaaS « Infrastructure as a Service »

IaaS est considéré comme une abstraction d'un centre de données. Les utilisateurs peuvent démarrer ou arrêter des serveurs virtuels dans ces centres de données à la demande sans se soucier de la machine physique ou des coûts de gestion qui sont liés (remplacement de matériel, climatisation, électricité, etc.). Ils gèrent les systèmes d'exploitation, les bases de données, les applications, les fonctions et toutes les données. En conséquence, il aura générale-

ment plus de contrôle et de flexibilité que dans d'autres modèles de services. Les fournisseurs de services Cloud possèdent et gèrent le matériel qui exécute la pile logicielle. Cela inclut les serveurs, les réseaux et le stockage [8],[9].

Cette approche nécessite de conserver les mêmes compétences informatiques au sein de l'entreprise que les solutions de serveur locales traditionnelles. Les utilisateurs doivent disposer de suffisamment d'experts en ressources humaines pour gérer eux-mêmes le système et s'assurer que les applications installées sur les machines virtuelles dans le Cloud fonctionnent normalement [4].

I.4.1.1 Exemples d'IaaS

- **Amazon Elastic Compute Cloud**¹ : Est un service Web qui fournit une capacité de calcul sécurisée et redimensionnable dans le Cloud. Destiné aux développeurs, il est conçu pour faciliter l'accès aux ressources de Cloud computing à l'échelle du Web.
- **OpenStack**² : Conçu pour configurer et gérer des grands réseaux des VMs. Il fournit des services fiables et évolutifs adaptés à une large classe de centres de données. En outre, il définit des API spécifiques pour permettre aux utilisateurs de créer, lancer et d'interagir avec leurs instances facilement et en toute sécurité.
- **OpenNebula**³ : Fournit des services d'IaaS entièrement open source conçus pour répondre aux besoins des entreprises (requirements of business) dans de nombreux secteurs, telles que l'hébergement web, télécommunications, administration en ligne.

I.4.2 PaaS « Platform as a Service »

Les fournisseurs PaaS fournissent une infrastructure de haut niveau pour les logiciels gérés, où les clients poussent leurs applications existant vers le Cloud ou utilisent des outils fournis par des fournisseurs pour développer de nouvelles applications, ils ne se soucient plus de la technologie pour déployer, mettre à jour et licencier les logiciels ou exécuter leurs applications. Les clients ne peuvent pas contrôler ces infrastructures car elles sont abstraites sous la plate-forme. Les services de plate-forme sont utilisés dans des domaines spécifiques, tels que

1. <https://aws.amazon.com>

2. <https://www.openstack.org>

3. <https://opennebula.io>

le développement d'applications Web et s'appuyant sur des langages de programmation. Les clients disposent d'un environnement distinct pour tester et développer ou déployer leurs applications [9].

L'une des faiblesses du PaaS réside dans les limites des technologies que les fournisseurs de services peuvent utiliser pour le développement d'applications, par exemple le nombre de langages de programmation utilisés est limité. Par conséquent, si les clients souhaitent changer de fournisseur PaaS, ils doivent réécrire toutes leurs applications, selon le fournisseur PaaS attendu. En effet, les clients ne peuvent pas contrôler l'évolution des services virtuels de leurs fournisseurs. Si les fournisseurs décident de mettre fin au développement de leurs services pour diverses raisons, des problèmes peuvent survenir [10].

I.4.2.1 Exemples PaaS

- **Amazon Web Services (AWS)**⁴ : Permet aux développeurs d'héberger, de déployer et gérer des applications dans le nuage.
- **Google AppEngine**⁵ : Est une plateforme de conception et d'hébergement d'application web basées sur les serveurs de Google.
- **Microsoft Azure**⁶ : Microsoft offre des services de plate-forme PaaS à l'aide de Windows Azure.

I.4.3 SaaS « Software as a Service »

NIST définit SaaS comme une capacité fournie au consommateur d'utiliser les applications du fournisseur s'exécutant sur une infrastructure Cloud. Les applications sont accessibles à partir de divers périphériques clients via une interface client léger, telle qu'un navigateur Web ou une interface de programme. Le consommateur ne gère pas le contrôle de l'infrastructure Cloud sous-jacente, y compris le réseau, les serveurs, les systèmes d'exploitation, le stockage ou même les capacités d'application individuelles, à l'exception peut-être de paramètres de configuration d'application limités et spécifiques à l'utilisateur [10].

Le SaaS assure la continuité de service en assurant la surveillance et l'intervention en cas

4. <https://aws.amazon.com>

5. <https://cloud.google.com/appengine>

6. <https://azure.microsoft.com/>

de panne. Les clients accèdent aux applications en mode hébergé sans se soucier de l'infrastructure matérielle ou des plates-formes logicielles. Ils n'achètent plus de logiciels, mais consomment à la demande. Le logiciel est hébergé par le fournisseur dans son propre centre de données. Les clients se déchargent complètement de la partie opérationnelle de l'environnement système qui leur permet de se concentrer davantage sur leur coeur de métier ou sur des projets innovants à haute valeur ajoutée [11].

Cependant, le modèle SaaS présente également quelques défauts majeurs, qui deviennent parfois des barrières à l'entrée pour certains utilisateurs. Confier des données à des machines virtuelles détenues par des fournisseurs de services soulève des problèmes supplémentaires de confidentialité et de sécurité. De plus, les clients sont limités aux services fournis par le fournisseur de services. En effet, les services et les fonctions de l'application sont généralement prédéterminés par le fournisseur de services. Par conséquent, toute particularité imprévue ne peut être développée que par le fournisseur [10].

I.4.3.1 Exemples de SaaS

- **Les services de stockages en ligne** : Le principal avantage du stockage dans le Cloud (ou stockage en ligne) est que vous pouvez obtenir des documents à partir de n'importe quel appareil connecté à Internet. Voici quelque service de stockage en ligne :

Mega⁷ :

- 20 Go d'espace.
- Chiffrage via votre mot de passe.
- Mega n'a pas accès à vos données.

iCloud⁸ :

- 5 Go.
- obligation d'avoir un iPhone ou un iPad avec iOS5.

Google Drive⁹ :

- 15 Go d'espace.
- Stockage illimité de photos.

PCloud¹⁰ :

7. <https://mega.io>

8. <https://www.apple.com/fr/icloud>

9. <https://www.google.com/intl/fr/drive>

10. <https://www.pcloud.com>

- 10 Go d'espace.
- Cryptage AES 256 bits.
- **Les applications en ligne** : Comme le stockage en ligne, le Cloud offre la possibilité d'utiliser des programmes distants directement sur internet via un navigateur Web. Voici quelques applications en ligne :
 - Pixlr**¹¹ : Un éditeur de photos en ligne qui vous permet d'éditer des photos et de créer de superbes designs gratuitement directement depuis le navigateur web.
 - PDF Online**¹² : Fournit des outils de traitement PDF gratuits (fusionner, convertir, éditer, compresser des PDF, etc.).
 - Audiotool**¹³ : Est un puissant studio de production musicale en ligne, directement depuis le navigateur web.

I.4.4 SaaS vs PaaS vs IaaS : Facilité de gestion vs Contrôle complet

La plupart des organisations aujourd'hui utilisent généralement les trois services, en combinaison avec l'informatique traditionnelle [12].

De toute évidence, la solution en tant que service choisi par le client dépend d'abord des fonctionnalités dont le client a besoin et des connaissances professionnelles des employés. Par exemple, les organisations qui ne disposent pas d'une expertise informatique interne pour configurer et exploiter des serveurs distants ne sont pas adaptées à l'IaaS, et les organisations qui n'ont pas d'équipe de développement n'ont pas besoin de PaaS. Mais dans certains cas, l'un des trois modèles « en tant que service » fournira une solution viable. Dans ces situations, les organisations comparent souvent les alternatives en fonction de la facilité de gestion qu'elles offrent et du contrôle qu'elles abandonnent [12].

La figure I.1 illustre les responsabilités de gestion du client et du fournisseur de Cloud pour chaque service Cloud :

11. <https://pixlr.com>

12. <https://pdf.online>

13. <https://www.audiotool.com>

	Traditionnel IT	IaaS	PaaS	SaaS
Applications	Vous avez le contrôle	Assuré par le prestataire	Assuré par le prestataire	Assuré par le prestataire
Data	Vous avez le contrôle	Assuré par le prestataire	Assuré par le prestataire	Assuré par le prestataire
Runtime	Vous avez le contrôle	Assuré par le prestataire	Assuré par le prestataire	Assuré par le prestataire
Middleware	Vous avez le contrôle	Assuré par le prestataire	Assuré par le prestataire	Assuré par le prestataire
OS	Vous avez le contrôle	Assuré par le prestataire	Assuré par le prestataire	Assuré par le prestataire
Virtualisation	Vous avez le contrôle	Assuré par le prestataire	Assuré par le prestataire	Assuré par le prestataire
Serveurs	Vous avez le contrôle	Assuré par le prestataire	Assuré par le prestataire	Assuré par le prestataire
Stockage	Vous avez le contrôle	Assuré par le prestataire	Assuré par le prestataire	Assuré par le prestataire
Réseau	Vous avez le contrôle	Assuré par le prestataire	Assuré par le prestataire	Assuré par le prestataire

■ Assuré par le prestataire
■ Vous avez le contrôle

FIG. I.1 : Les services de Cloud Computing [12].

I.4.5 Autres types de Services Clouds

- **NaaS « Network as a Service »**

NaaS est un modèle Cloud qui permet aux utilisateurs d'exploiter facilement le réseau et d'obtenir les résultats qu'ils attendent sans avoir à posséder, créer ou entretenir leur propre infrastructure, mettre en œuvre des décisions de transfert personnalisées en fonction des exigences de l'application, telles que traiter des paquets au cours du processus, en choisissant de modifier la charge utile ou de créer dynamiquement de nouveaux paquets, au lieu d'utiliser leur propre WAN qui nécessite une maintenance constante et provoque souvent un goulot d'étranglement du trafic réseau [13], [14].

- **DBaaS « Database as a Service »**

DBaaS (également connu sous le nom de service de base de données) est un service de Cloud Computing qui permet aux utilisateurs d'accéder et d'utiliser des systèmes de base de données Cloud sans avoir à acheter et configurer leurs propres matériels, installer leurs propres logiciels de base de données ou gérer la base de données eux-mêmes. Le fournisseur de Cloud est responsable de tout, des mises à niveau régulières aux sauvegardes, pour s'assurer que le système de base de données reste disponible [15].

I.5 Composants fonctionnels de Cloud Computing

- **Prestataire de service Cloud (Cloud Providers « CPs »)** : C'est une entité qui gère le serveur de stockage Cloud (CSS pour Cloud Storage Server) et fournit des services Cloud (par exemple, SaaS, PaaS et IaaS) aux consommateurs, aux courtiers de services ou aux revendeurs [16].
- **Client/propriétaire (Cloud Consumers « CCs »)** : C'est une entité qui a un grand nombre de fichiers de données stockés dans le Cloud et qui s'appuie sur le Cloud pour la maintenance et le calcul des données, il peut être soit un consommateur individuel ou une organisation [16].
- **Utilisateur** : C'est une unité qui est inscrit sur le propriétaire et utilise les données de celui-ci stockées sur le Cloud. L'utilisateur peut être un propriétaire lui-même [16].
- **Courtier (Cloud Brokers « CBs »)** : C'est un intermédiaire entre les consommateurs de Cloud et les fournisseurs qui disposent leurs services. Généralement, deux types de courtiers sur un marché Cloud peuvent être distingués. Premièrement, il existe des courtiers qui se concentrent sur la négociation des relations entre les consommateurs et les fournisseurs sans posséder ni gérer l'infrastructure Cloud. Ils fournissent, par exemple, des services de conseil aux potentiels CCs pour déplacer leurs ressources informatiques dans un Cloud adapté. Deuxièmement, il existe des courtiers qui ajoutent des services supplémentaires à l'infrastructure / plate-forme / logiciel d'un CPs pour améliorer et sécuriser l'environnement Cloud pour les consommateurs. Par exemple, un courtier peut offrir un service de gestion des identités et des accès en plus des offres de services de base du CP aux consommateurs [16].
- **Revendeur (Cloud Resellers « CRs »)** : Les revendeurs fournissent des services pour le compte d'un fournisseur Cloud. Ils peuvent devenir un facteur important sur le marché du Cloud lorsque les CP étendent leurs activités à de nouveaux marchés, par exemple à travers les continents. Les CP peuvent choisir des sociétés de conseil informatique locales ou des revendeurs de leurs produits existants pour agir en tant que revendeurs de leurs produits basés sur le Cloud dans une région particulière [16].
- **Auditeur (Cloud Auditors « CAs »)** : Les auditeurs effectuent des évaluations indépendantes d'autres entités (y compris les opérateurs Cloud) sur les performances et la sécurité des services Cloud, les opérations du système d'information et la mise en œuvre du Cloud. Par exemple, les auditeurs Cloud peuvent évaluer les contrôles de

sécurité dans le système d'informations pour déterminer le niveau de contrôle qui est mis en œuvre correctement ou non. Sur la base de l'évaluation, ils délivrent un certificat d'audit spécifique, ce qui est extrêmement important pour les consommateurs [16].

- **Carriers (Cloud Carriers « CCas »)** : Ces entités intermédiaires assurent une fourniture de service transparent en fournissant des connexions entre les entités Cloud. Par exemple, CCas fournit un réseau, des télécommunications et d'autres équipements pour accéder aux services Cloud. Les opérateurs de réseaux et de télécommunications entrent également dans cette catégorie car ils veillent à ce que les services soient distribués via leurs réseaux [16].

I.6 Taxonomie du marché du Cloud Computing

La taxonomie du Cloud décrit les entités du marché du Cloud. Les entités sont classées en fonction de leur rôle sur le marché du Cloud. Ensuite, il classe chaque entité selon le type de service qu'elle fournit ou consomme. La taxonomie du Cloud présentée dans la figure (I.2) illustre clairement la structure diversifiée du marché du Cloud Computing.

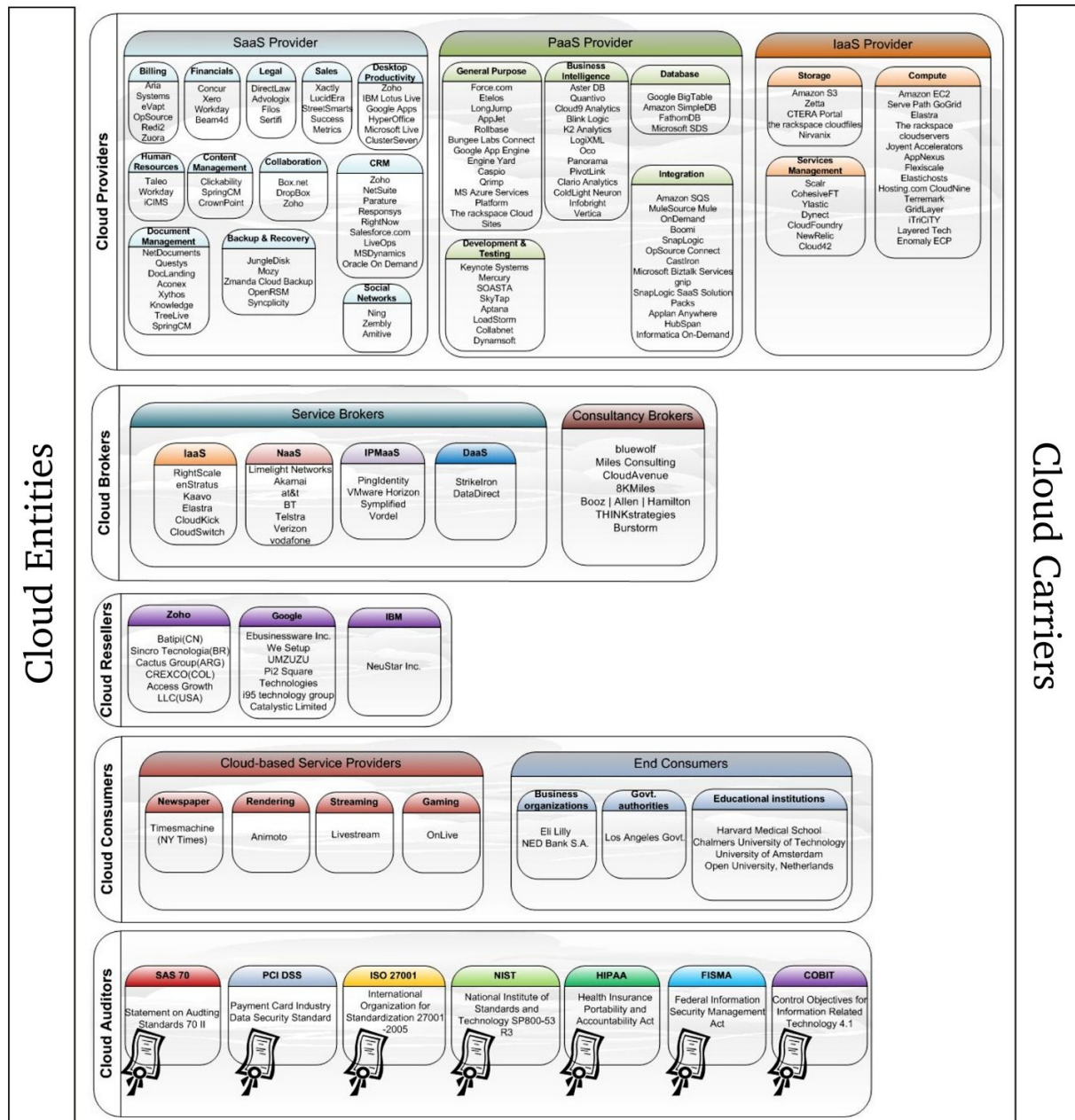


FIG. I.2 : Taxonomie du marché du Cloud Computing [16].

Une brève discussion sur la classification est donnée ci-dessous :

- Dans la première ligne, les CP sont répertoriés selon les modèles de prestation de services (Saas, Paas et Iaas.). Dans chaque modèle de prestation de services, il existe divers types de services offerts par les CP. Par exemple, on peut voir que Zoho fournit un type spécifique de produits SaaS, la productivité de bureau, mais ne fait pas partie des entreprises fournissant, par exemple, des réseaux sociaux en tant que service [16].
- Dans la deuxième ligne, les CBs sont répertoriés selon le type de service qu'ils ont négocié. Par exemple, Right Scale fournit une plate-forme de gestion Cloud qui permet

aux consommateurs de gérer de manière flexible l'infrastructure hébergée par différents fournisseurs d'infrastructure Cloud [16].

- Dans la troisième ligne, les CRs sont répertoriés sous les CPs respectifs pour les offres dont ils agissent en tant que revendeurs [16].
- Dans la quatrième ligne, CCs sont représentées selon le modèle économique qu'elles suivent. Par exemple, Animoto, un fournisseur de services de rendu vidéo hébergé dans le Cloud d'Amazon (agissant ainsi en tant que fournisseur de services pour ses clients, mais consommant des services informatiques basés sur le Cloud d'Amazon), propose des services de rendu à ses consommateurs finaux, pour la création de vidéos diapositives à partir des images données. Les consommateurs finaux consomment des services Cloud, mais ne fournissent pas de services Cloud à d'autres clients dans le cadre de leur modèle commercial principal. Par exemple, les établissements d'enseignement (l'Université de technologie Chalmers ou l'Université d'Amsterdam), les organisations commerciales (Eli Lilly) et les autorités gouvernementales (le gouvernement de la ville de Los Angeles) utilisent des ressources Cloud pour leurs besoins informatiques mais ne pas vendre les services en dehors de leurs limites ou aux consommateurs en dehors des organisations [16].
- La cinquième ligne répertorie une série de normes (ou services) d'audit Cloud basées sur les ressources liées à l'audit de la Cloud Security Alliance [16].

I.7 Accord de niveau de service (SLA)

Un Accord de niveau de service ou en anglais Service Level Agreement est un accord entre deux ou plusieurs parties, l'une est le fournisseur de services et l'autre est le consommateur, auquel ces deux derniers arrivent après des négociations sur la qualité de service, la supervision des ressources et le contrôle des coûts attendus par les consommateurs, plutôt que toute autre norme que le fournisseur s'engage à respecter [17].

I.7.1 Composants essentiels d'un SLA

Le contenu typique d'un SLA est présenté dans le travail de [17], et est présenté comme suit :

- **Définition des services** : Il s'agit de la partie la plus critique de l'accord car elle décrit les services et la manière dont ils sont fournis.

- **Gestion des performances** : chaque service doit être mesurable et les résultats doivent être analysés et rapportés. Les paramètres à utiliser doivent être spécifiés dans l'accord lui-même.
- **Garanties et recours** : Cette partie du SLA couvre généralement les sujets clés suivants : La compensation pour la qualité du service et les recours en cas de rupture de contrat, etc.
- **Sécurité** : Le fournisseur doit respecter et se conformer aux politiques et procédures de sécurité du client.
- **Gestion des problèmes** : Le but de la gestion des problèmes est de minimiser l'impact négatif des incidents et des problèmes. Cela spécifie généralement qu'il doit y avoir un processus approprié pour gérer et résoudre les incidents, et qu'il doit également y avoir des activités préventives pour réduire l'occurrence des incidents.

I.7.2 Caractéristiques du SLA

- **Améliorer la satisfaction du consommateur** : Un SLA aide les fournisseurs à se concentrer sur les besoins des consommateurs et à garantir que les efforts vont dans la bonne direction [18].
- **Qualité de service modifiée** : Les exigences de QOS sont généralement déterminées dans les accords de niveau de service. Le SLA consiste en un ensemble d'accords contractuels entre les fournisseurs de services et les consommateurs, qui stipulent la nature, la valeur et la finalité des services à fournir. Avant que le fournisseur de services ne soit pénalisé, le SLA peut également déterminer le pourcentage de violations qui sont acceptables dans un délai prédéfini [18].
- **Améliorer la relation entre les deux parties** : Un SLA sain spécifie la stratégie de compensation et de punition pour le service. Les clients peuvent observer les services en fonction des objectifs de niveau de service (SLO) définis dans le SLA [18].

I.8 Fondements technologiques du Cloud Computing

Les technologies et les infrastructures de base nécessaires pour construire un Cloud, indépendamment du son type sont :

I.8.1 Centre de données « Data Center »

Les Clouds ont besoin de serveurs sur le réseau, et ils ont besoin d'une maison (station). Cette maison physique et tous les équipements en font un data center. Un centre de données est une installation physique qui constitue le cœur de l'informatique d'entreprise [19]. Il héberge les éléments suivants : [19]

- Systèmes informatiques d'entreprise.
- Équipement de réseau et matériel connexe requis pour assurer une connexion continue entre le système informatique et Internet ou d'autres réseaux de l'entreprise.
- Alimentations électriques et sous-systèmes, commutateurs électriques, générateurs de secours et contrôles environnementaux (tels que l'équipement de climatisation et de refroidissement des serveurs) qui protègent le matériel du centre de données et lui permettent de fonctionner.

Le centre de données est au cœur des opérations informatiques de l'entreprise. C'est le référentiel des systèmes les plus critiques, où la plupart des données commerciales sont stockées, traitées et diffusées aux utilisateurs.

Le centre de données est toujours composé de trois éléments principaux qui sont essentiels au fonctionnement de l'environnement informatique : [19]

- **Calcul** : La mémoire et la puissance de traitement nécessaires pour exécuter des applications généralement fournies par les serveurs d'entreprise.
- **Stockage** : Le centre de données dispose d'une unité de stockage principale et d'une unité de stockage de sauvegarde. Il peut s'agir de disques durs ou même de lecteurs de bande, mais les meilleures installations sont généralement des matrices de mémoire flash à 100%.
- **Réseaux** : Ils contiennent toutes sortes d'équipements réseau, des routeurs et commutateurs aux contrôleurs et pare-feu.

I.8.2 Virtualisation

La virtualisation est l'une des principales technologies qui ont aidé à démarrer le Cloud Computing. La virtualisation est une méthode d'exécution de plusieurs systèmes d'exploitation virtuels indépendants sur un seul ordinateur physique. La création et la gestion de

machines virtuelles sont souvent appelées virtualisation de plate-forme (Platform virtualisation). La virtualisation de plate-forme est effectuée sur un ordinateur donné (plate-forme matérielle) à l'aide d'un logiciel appelé programme de contrôle. Le programme de contrôle crée un environnement simulé qui permet l'utilisation de logiciels hébergés spécifiques à l'environnement virtuel, parfois appelés logiciels invités (Guest Software) [20]. Par conséquent, la virtualisation implique trois composants principaux :

- Un système d'exploitation principal installé sur la machine physique, appelé système hôte, car il joue le rôle d'hôte à d'autres systèmes d'exploitation.
- Un hyperviseur un outil de virtualisation installé sur le système hôte qui fournit l'environnement dans lequel différentes machines virtuelles s'exécutent. A l'aide de cet hyperviseur, nous pouvons diviser les ressources physiques sous-jacentes en plusieurs ressources virtuelles, également appelées VM. Chaque VM est comme un conteneur virtuel dédié avec un système d'exploitation et une application spécifique [20].
- Un système d'exploitation installé dans une VM, appelé système invité, qui fonctionne indépendamment des autres systèmes invités dans d'autres VMs [5].

I.9 Modèles de déploiement du Cloud Computing

Le modèle de déploiement définit qui gère les ressources virtuelles du Cloud et qui possède l'infrastructure physique du Cloud. On peut distinguer quatre principaux modèles de déploiement pour ces Clouds :

I.9.1 Cloud public « Public Cloud »

Un Cloud public peut être consulté par tout abonné disposant d'une connexion Internet et un accès à l'espace Cloud et est géré par une organisation. L'organisation peut être une entreprise (comme Google), un département académique ou gouvernemental. Le Cloud public ne signifie pas que les données des utilisateurs sont visibles publiquement, car les fournisseurs de Cloud public offrent généralement des mécanismes de contrôle d'accès pour les utilisateurs [21]. Avec ce modèle, les clients n'ont ni visibilité ni contrôle sur l'emplacement de l'infrastructure. Il est important de noter que tous les clients sur le Cloud public partagent le même pool d'infrastructure avec une configuration limitée, des protections de sécurité et des écarts de disponibilité. L'avantage de cette architecture est qu'elle est facile à mettre en

place et que le coût est relativement raisonnable. La charge du matériel, des applicatifs, de la bande passante étant couverte par le fournisseur. Il n'y a pas de gaspillage de ressources car le client ne paye que ce qu'il consomme [22].

I.9.2 Cloud privé « Private Cloud »

Contrairement au Cloud public, le Cloud privé n'est pas disponible pour le grand public, il est propre à chaque organisation.

Les Clouds privés mettent l'ensemble des ressources à la disposition exclusive d'une seule entreprise. Les services de ces Clouds sont dédiés à répondre aux besoins spécifiques des entreprises clientes. Ce type de Cloud offre aux entreprises des services qui leur permettront d'optimiser et de mieux maîtriser l'utilisation de leurs ressources informatiques. Les problèmes d'intégration et les exigences de sécurité des données et applications critiques sont l'une des raisons pour lesquelles les gens adoptent les clouds privés. Avec les Clouds privés, les utilisateurs peuvent contrôler leurs applications et leurs données plus que d'autres types de Cloud [22].

I.9.3 Cloud Communautaire « Community Cloud »

C'est un modèle de déploiement qui est en cours de mise en œuvre rapide. Conceptuellement, il réside quelque part entre un Cloud privé et un Cloud public [23]. Dans un Cloud communautaire, l'infrastructure est partagée entre deux ou plusieurs organisations indépendantes ayant les mêmes préoccupations, besoins et exigences spécifiques, selon le type de réglementation restreinte pour permettre la collaboration sur le même projet. C'est une solution sur mesure pour différents segments de marché [6].

I.9.4 Cloud hybride « Hybrid Cloud »

Est un environnement composé de multiples prestataires internes et externes. Aujourd'hui, de nombreuses entreprises utilisent des Clouds publics et privés. Un Cloud hybride est essentiellement une combinaison d'au moins deux Clouds (Cloud basé sur plusieurs nuages), où le Cloud est un mélange de Clouds public, privé ou communautaire, Les différents Clouds qui le composent restent des entités distinctes, soit au travers de standards, soit au travers

de technologies propriétaires qui permettent la portabilité des applications déployées sur différents Clouds. Ce modèle vise à remédier aux limitations de chaque approche [5].

I.10 Avantages du Cloud Computing

Comme chaque nouvelle tendance technologique a ses avantages et ses inconvénients, parmi les nombreux avantages de Cloud nous avons listé les points suivants :

I.10.1 Flexibilité

- **Options de stockage** : Les utilisateurs peuvent choisir parmi des offres de stockage public, privé ou hybride en fonction de leurs besoins de sécurité [24].
- **Options de contrôle** : Les entreprises peuvent choisir leur niveau de contrôle via l'option « as a service » : SaaS, PaaS et IaaS ou autres services [24].
- **Choix d'outils** : Les utilisateurs disposent d'un menu préconfiguré d'outils et de fonctions pour créer des solutions répondant à leurs besoins spécifiques [24].

I.10.2 Déploiement mondial en quelques minutes

Avec l'aide du Cloud, vous pouvez étendre vos activités à de nouvelles régions et la promouvoir à l'international en quelques minutes. Par exemple, AWS dispose d'infrastructures dans le monde entier. Par conséquent, vous pouvez déployer votre application dans différentes parties du monde en quelques clics [25].

I.10.3 Efficacité

- **Accessibilité** : Les applications et les données dans le Cloud sont accessibles depuis presque tous les appareils connectés à Internet [24].
- **Sécurité des données** : Les pannes matérielles n'entraînent pas de perte de données grâce aux sauvegardes réseau [24].
- **Réduction des coûts** : Le Cloud vous permet de remplacer vos investissements en capital (comme les centres de données et les serveurs physiques) à des coûts variables, et de ne payer que pour les services informatiques que vous utilisez [25].

I.10.4 Agilité

Le Cloud vous donne un accès facile à un large éventail de technologies. Vous pouvez utiliser rapidement les ressources dont vous avez besoin, des services d'infrastructure tels que l'informatique, le stockage et les bases de données à l'Internet des objets, l'apprentissage automatique, les lacs de données et l'analyse. Vous pouvez déployer des services techniques en quelques minutes et passer de l'idée à la mise en œuvre plus rapidement [25].

I.11 Limites du Cloud Computing

Parmi les nombreux limites du Cloud, nous avons évoqué les points suivants :

I.11.1 Sécurité

La sécurité et la confidentialité sont les principales préoccupations concernant le Cloud Computing. Les données sont hébergées à l'extérieur de l'entreprise. Cela peut donc poser un risque potentiel pour l'entreprise de voir ses données stockées dans un serveur virtuel qui rend la responsabilité de la sécurité des données difficile à déterminer et même les utilisateurs peuvent se sentir mal à l'aise de remettre leurs données à un tiers [26].

I.11.2 Connectivité

Le système Cloud est entièrement dépendant de la connexion Internet. Si la connexion Internet ne convient pas, la connexion au serveur n'est pas autorisée [27].

I.11.3 Problèmes de transfert de données

Les services Cloud dépendent d'Internet, de sorte que la vitesse d'accès aux données peut être réduite car chaque fois qu'un utilisateur envoie une demande d'accès aux données, toutes les données sont extraites du Cloud et renvoyées à nouveau. Il est très différent du système informatique traditionnel, où les données sont accessibles localement et immédiatement [23].

I.11.4 Perte de données

Les services Cloud sont ciblés pour des attaques et donc ouverts à être infestés de bugs et Virus. Cela pourrait conduire à des pertes de données qui pourraient être très désastreuses pour les utilisateurs de Cloud [27].

I.12 Conclusion

Le Cloud Computing est un phénomène qui est largement discuté en raison de sa facilité d'exploitation des données.

Dans ce chapitre nous avons parlé des généralités du Cloud Computing, en citant tous les éléments qui permettent de bien comprendre le domaine et d'avoir une conception globale de l'environnement Cloud.

Dans le prochain chapitre, nous allons définir les concepts de la description, la découverte, la sélection et la composition des services de Cloud Computing.

Chapitre II

La découverte des services dans le Cloud

II.1 Introduction

Au cours du dernier chapitre, nous avons défini ce qu'est le Cloud Computing et quels sont ses composants, ses avantages et ses inconvénients.

Le développement rapide du Cloud Computing a attiré l'attention des chercheurs et des sociétés de IT. Selon [28], les études d'analyse commerciale de Gartner, les prévisions de revenus du marché du Cloud de 2019 à 2022 sont :

- IaaS : 81 milliards de dollars.
- PaaS : 72 milliards de dollars.
- SaaS : 140 milliards de dollars.

Les services Cloud publiés souffrent du problème d'hétérogénéité, où les services de la même fonctionnalité publiés à partir d'environnements hétérogènes par différents fournisseurs de Cloud ont des caractéristiques, des descriptions différentes, tel que chaque fournisseur de services Cloud utilise son propre SDL. De plus, la croissance exponentielle du nombre de services et les connaissances insuffisantes des utilisateurs sur la technologie Cloud posent certains problèmes qui représente directement ou indirectement les principaux problèmes de nombreux défis liés au Cloud. La découverte de services Cloud est l'un de ces défis qui sont directement touchés par ces problèmes. Donc la question est comment les utilisateurs peuvent-ils trouver tous les services offerts, par millions ou milliards des services et comment

ils font le choix le plus approprié. De plus comment composer les services pour la demande d'utilisateur dans le cas où il n'existe pas un service spécifié pour la demande d'utilisateur ?

Dans ce qui suit nous répondons aux questions suivantes :

- ✓ Qu'est-ce que la description de service ?
- ✓ Quels sont les langages de description ?
- ✓ Qu'est-ce que la découverte et la sélection ?
- ✓ Qu'est-ce que la composition des services ?

II.2 Description de service Cloud

Dans cette section, nous définissons la description de service et mentionnons les langages de description de service existantes.

II.2.1 Définition

La description permet de décrire les caractéristiques clés d'un service Cloud, pour rendre le service compréhensible aux systèmes utilisés, et pour faciliter plusieurs opérations sur le service [29].

Chaque fournisseur de Cloud a ses propres définitions, interprétations et types de valeur pour différents attributs de son services. Cela nécessite que chaque fournisseur utilise son propre SDL, en utilisant sa grammaire et sa sémantique sous-jacentes associées au modèle et aux normes [30].

II.2.2 Aspects de description de service Cloud

1. Contrainte fonctionnelles

Elle précise les opérations que le service peut fournir, autrement dit c'est la fonction de service ou ce qu'il va faire et elle participe à la définition des besoins du client. Elle peut inclure les tâches à exécuter, le domaine de service, etc [31].

2. Contrainte non fonctionnelles

Afin de quantifier un service, il existe certaines mesures. Celles-ci sont représentées par des contraintes non fonctionnelles, également appelées : Qualité de Service. Elles définissent les capacités de service à fonctionner dans de bonnes conditions, et elles doivent toujours être décrites en termes clairs, précis et sans ambiguïté. De plus, ces valeurs permettent de sélectionner des services liés à la demande de l'utilisateur. Les caractéristiques considérées pour déterminer la qualité du service diffèrent évidemment selon le service rendu [32]. Dans cette section nous allons présenter quelques critères de QoS :

1. Le coût

Il s'agit du coût d'accès et d'utilisation du service que le consommateur du service doit payer. Il dépend du nombre de tâches que l'utilisateur du service doit effectuer, et diffère d'un fournisseur à un autre [33], [34].

2. Fiabilité

Elle fait référence à la capacité du service à fonctionner correctement et de manière cohérente conformément aux exigences spécifiées dans le SLA. Elle est mesurée en termes de défaillances de transaction par année ou par mois [35], [34]. La fiabilité est calculée selon l'équation suivante :

$$Fiabilité = \left(\frac{1 - Nombre\ d'échec}{n} \right) P_{mttf} \quad (II.1)$$

Nombre d'echec est le nombre d'utilisateurs qui ont subi un échec dans un intervalle de temps inférieur à celui prévu par le fournisseur du Cloud, *n* est le nombre d'utilisateur, Où *P_{mttf}* représente le temps d'échec estimé qui est défini par le fournisseur de services.

3. Disponibilité

La disponibilité réfère à la capacité d'un système de masquer ou de corriger ses erreurs de façon afin que la durée cumulative d'absence de service ne dépasse pas une valeur spécifiée au cours d'un intervalle de temps donné par le fournisseur [34], [35]. La disponibilité est calculée par l'équation suivante :

$$Disponibilité = \frac{MTBF}{(MTBF + MTTR)} \quad (II.2)$$

MTBF se réfère au temps moyen de service avant l'état d'échec, et *MTTR* est le temps moyen de réparation.

4. Sécurité

La sécurité est la capacité d'un système à protéger les données et les informations contre tout accès non autorisé. Il s'agit donc de l'ensemble des exigences de sécurité lors de la fourniture d'un service [34].

Les contraintes non-fonctionnelles de service peuvent être soit positives soit négatives.

- **Contraintes non-fonctionnelles positives** : Sont les attributs qu'il faut maximiser (disponibilité, fiabilité, sécurité, etc.).
- **Contraintes non-fonctionnelles négatives** : Sont les attributs qu'il faut minimiser (temps de réponse, le coût, etc.).

CSD définit les contraintes fonctionnelles et la qualité des services pour les utilisateurs experts (tels que les développeurs d'applications) d'un point de vue technique et opérationnel. Cependant, dans certains cas, il inclut également une perspective commerciale pour attirer les utilisateurs professionnels (utilisateurs de services non techniques) tels que les utilisateurs commerciaux. En particulier, le modèle SLA est utilisé pour décrire la fonctionnalité du service d'un point de vue non technique (c'est-à-dire commercial) [30].

II.2.3 Les langages de description de service

Dans la littérature, plusieurs langages sont proposés pour décrire les services. Ces modèles sont utilisés pour présenter les services de façon compréhensible aux systèmes utilisés, ils sont regroupés dans deux catégories : **description syntaxique** et **description sémantique**.

- **description syntaxique** signifie que le service est interprétable par l'utilisateur et qu'il est invocable de manière statique.
- **description sémantique** signifie que la machine peut interpréter et invoquer le service (recherche et invocation de service se fait dynamiquement).

Les paramètres considérés dans la description des services diffèrent en fonction du type de

service décrit. Dans ce qui suit, nous présenterons les langages utilisés pour les services Web et les services Cloud.

Parmi ces langages, on discute :

- **WSDL**¹ : Est une spécification basée sur XML qui permet aux fournisseurs de spécifier les fonctions et les caractéristiques opérationnelles des services Web, telles que les interfaces, les opérations, les protocoles de liaison et les adresses de point de terminaison.
- **USDL** [36] : Il a été développé pour définir un langage général pour spécifier à la fois les services commerciaux et techniques. Les services techniques sont considérés comme les spécifications de WSDL ou toute autre spécification technique, tandis que les services commerciaux sont définis comme des activités commerciales fournies par le fournisseur de services pour créer de la valeur pour le consommateur. Par conséquent, il permet aux fournisseurs de services de publier un service en décrivant ses caractéristiques générales via USDL, ce qui contribue également à une meilleure découverte et sélection du service.
- **SAWSDL**² : Le langage SAWSDL permet d'annoter les descriptions WSDL pour permettre une description sémantique de ses éléments.
- **OWL**³ : Est un langage de description d'ontologies conçu pour la publication et le partage d'ontologies sur le web sémantique. Il s'appuie sur RDFS afin de l'enrichir en définissant un vocabulaire plus complet pour la description d'ontologies complexes.
- **OWL-S** [37] : La définition de nouvelles ontologies pour la description des services Web, tel que OWL-S, qui est basé sur le langage OWL. Ce dernier étend RDF par la définition de notions telles que les propriétés d'équivalence, de contraire et d'identiques entre deux ressources. OWL-S permet de décrire ce que fait le service Web, comment l'utiliser et comment y accéder.

II.3 Découverte de service Cloud

Dans cette section, nous définissons c'est quoi la découverte, les environnements et les méthodes de découverte de service Cloud.

1. <https://www.w3.org/TR/2001/NOTE-wsdl-20010315>

2. <https://www.w3.org/2001/sw/wiki/SAWSDL>

3. <https://www.w3.org/OWL>

II.3.1 Définition

La découverte des services Cloud est une opération qui permet de localiser une description de service Cloud exploitable et inconnue par les machines sur un réseau, et qui répond à certains critères fonctionnels. Il s'agit de faire correspondre entre les critères avec un ensemble de descriptions des services. L'objectif est de trouver un ensemble des services appropriés liés aux critères fonctionnels [38].

Un système de découverte devrait pouvoir identifier une similitude entre les spécifications des services utilisateurs et les offres fournis par les fournisseurs de services. Il existe deux mécanismes de découverte :

- **La découverte syntaxique :**

Généralement basé sur une comparaison entre les mots-clés extraits des requêtes des utilisateurs et les mots-clés extraits des descriptions de services. Le principal inconvénient de la découverte syntaxique est qu'elle ne prend pas en compte les aspects sémantiques de la requête.

- **La découverte sémantique :**

Elle consiste à utiliser différents concepts trouvés dans l'ontologie ou des ressources sémantiques externe pour calculer le degré de correspondance entre les mots-clés de requête et les mots-clés de description de service [38].

II.3.2 Environnement de découverte

Les solutions de découverte de services Cloud peuvent être divisées en quatre types :

- **Approches sémantiques centralisées**

L'approche sémantique centralisée dépend d'un nœud central qui fournit une vue complète de tous les services Cloud proposés sur le marché [39]. Un dépôt est un élément essentiel de cette approche pour maintenir les attributs des services Cloud. Ce dépôt permet généralement aux fournisseurs ou développeurs de Cloud d'enregistrer leur liste de services ainsi qu'une description du service qu'ils proposent. De plus, il offre aux consommateurs du Cloud un emplacement unique à partir duquel ils peuvent sélectionner leur service. Le dépôt fonctionne comme un médiateur entre les fournisseurs de

Cloud et les clients, et il aide à mettre en place l'association entre eux. Dans ce type d'approche de découverte, la description sémantique des services Cloud est représentée par une ontologie. Une ontologie de demandeur de service en Cloud est utilisée pour demander l'exécution de l'application demandant des ressources, et une ontologie de fournisseur est utilisé pour allouer des ressources sous la forme de services [34].

- **Approches Syntaxique centralisées**

Dans ce type d'approche, un registre agit comme un nœud central pour conserver toutes les informations pour les services Cloud existants, et les consommateurs peuvent sélectionner les services en fonction de leurs besoins. Cependant, l'ontologie n'est pas utilisée dans le processus de découverte ni dans la représentation des services. Au lieu de cela, des techniques d'apprentissage automatique sont appliquées pour mettre en œuvre le processus de découverte, et les services sont représentés en utilisant des langages de texte ou de description tels que WSDL pour décrire les services Cloud et leurs fonctionnalités spéciales [34].

- **Approches sémantiques décentralisées**

L'approche sémantique décentralisée repose sur l'établissement d'un lien direct entre le consommateur de Cloud et le fournisseur de services. Dans cette approche, le fournisseur de services publie ses informations de description de service sur Internet via sa page Web officielle. De plus, le fournisseur actuel décrit les informations de QoS fonctionnelles, telles que le prix et, par conséquent, le consommateur peut choisir un service approprié en fonction de ses exigences fonctionnelles [39]. Dans ce type d'approche, soit un Crawler privé est lancé pour trouver et rechercher les services Cloud proposés sur les sites Web de nombreux fournisseurs. De plus, l'ontologie est utilisée comme un outil de représentation des connaissances des concepts de recherche sous-jacents pour améliorer les performances des Crawler [34]. Une fois que le consommateur a trouvé les offres, une comparaison entre ces offres est effectuée manuellement par le consommateur lui-même. Bien que l'approche sémantique décentralisée permet toujours d'accéder à des informations de service mises à jour directement du fournisseur, trouver et comparer les offres de service est une tâche longue et difficile [39].

- **Approches Syntaxique décentralisées**

Dans cette approche, une connexion directe entre un consommateur de Cloud et un fournisseur est établie, de sorte que le consommateur peut rechercher les services en

fonction de ses besoins à travers un moteur de recherche qui fournit une description des services par les fournisseurs. De plus, cette approche n'utilise pas l'ontologie dans le processus de découverte ou de description de services Cloud [34].

II.3.3 Méthodes de découverte

Dans la littérature, les chercheurs ont appliqué différentes méthodes pour mettre en œuvre des systèmes de découverte de services Cloud. Ces méthodes peuvent être divisées en deux méthodes :

- **Approches fondées sur les agents**

Pour traiter la découverte des services Cloud, les agents peuvent être utilisés pour obtenir des informations de l'utilisateur. Cet agent implique la découverte des services Cloud demandés à partir du registre de services sémantiques à l'aide des informations fournies par le consommateur de services. Alors, le service est recherché selon les exigences définies par l'utilisateur [40]. L'agent reçoit la demande et effectue le jumelage et le raisonnement des relations entre les services Cloud identifiés. La description sémantique des fournisseurs de services Cloud et de leurs attributs est fournie via des ontologies de services. Dans la littérature, de nombreux chercheurs ont appliqué cette méthode pour aider à la découverte de services Cloud [34].

- **Approches intelligentes**

Dans cette approche de mise en œuvre d'une découverte de services en Cloud, l'exploration de données et les techniques d'apprentissage automatique sont utilisées. Des méthodes de classification sont utilisées pour déterminer le service Cloud approprié en fonction des besoins des utilisateurs. Ensuite, des méthodes de Cluster sont appliquées pour extraire les fonctionnalités du service Cloud identifié [34].

II.4 Sélection de service Cloud

Dans cette section, nous définissons c'est quoi la sélection des services Cloud et comment sélectionner le meilleur choix de service.

II.4.1 Définition

La sélection consiste à choisir le meilleur service en fonction de contraintes d'utilisateurs non fonctionnels parmi de nombreux choix de services.

Le fournisseur de services offre des services informatiques, qui comprend les attributs QoS. Cependant, différents fournisseurs de services peuvent fournir différents niveaux d'attributs de QoS, notamment la disponibilité, l'évolutivité, la flexibilité, la sécurité et une facturation précise. Cela pose un défi aux utilisateurs pour trouver des services qui répondent à leurs besoins.

L'un des principaux objectifs de la découverte de services est de fournir une comparaison raisonnable entre les services fournis par les fournisseurs afin que les utilisateurs puissent comparer et sélectionner des services qui répondent à leurs besoins en fonction de plusieurs spécifications prédéfinies.

Le processus de sélection est généralement basé sur la performance multiforme des services alternatifs et est donc modélisé comme un problème de décision multicritères (MCDM). Les clients peuvent déterminer le niveau d'importance de chaque aspect de la performance. Le résultat de la sélection peut ainsi refléter les besoins individuels du client.

II.4.2 Prise de décision multicritère MCDM

Plusieurs approches ont été proposées pour résoudre le problème de sélection de service, y compris l'analyse de décision multicritère MCDA ou MCDM.

MCDM consiste à prendre des décisions lorsque plusieurs critères (ou objectifs) doivent être pris en compte simultanément afin de classer ou de choisir entre des alternatives multiples avec plusieurs contraintes non fonctionnelles (QoS) [41].

MCDM implique quatre composants clés de base :

- Alternatives à classer ou à choisir entre.
- Critères d'évaluation et de comparaison des alternatives.
- Un poids représentant l'importance relative des critères.
- Les critères saisis par le client représentent son choix.

Un premier pas extrêmement important dans la prise de décision est de formuler le problème

de décision de manière formelle et rigoureuse. Donc il faut d'abord présenter la solution de sélection de service Cloud sous une forme mathématique généralisée et abstraite comme mentionnée dans [41] :

1. **Services** : Soit $S = \{S_1, S_2, \dots, S_s\}$ ensemble des services publié par les fournisseurs, tel que $s \geq 2$.
2. **Critères de performance** : Soit $C = \{C_1, C_2, \dots, C_c\}$ ensemble des critères de service utiles pour la sélection, tel que $c \geq 2$.
3. **Fonction de mesurent de performance** : Soit $F = \{F_1, F_2, \dots, F_f\}$, F_i représente la fonction d'évaluation de chaque critère C_i , tel que $f \geq 2$.
4. **Vecteur de description de service de fournisseur** : Soit $D = \{D_1, D_2, \dots, D_n\}$, un vecteur qui décrit un service S_i sous critère C_i tel que $D_i = F_i(S_i)$. Certains des critères peuvent être de nature qualitative et doivent être attribués à des valeurs quantitatives. De plus, ces valeurs doivent être normalisées pour éliminer les problèmes de calcul résultant d'une dissimilitude dans les unités de mesure. La procédure de normalisation est utilisée pour obtenir des unités sans dimension qui sont comparables, généralement compris entre 0 et 1. Les deux méthodes de normalisation les plus populaires sont : la normalisation linéaire et la normalisation vectorielle.
5. **Matrice de décision** : Les vecteurs de description des services D_i peuvent être combinés dans une matrice de décision M de dimension $l \times n$:

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdot & \cdot & \cdot & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdot & \cdot & \cdot & a_{2,n} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{l,1} & a_{l,2} & \cdot & \cdot & \cdot & a_{l,n} \end{pmatrix}$$

Chaque ligne de la matrice représente les valeurs des QoS pour chaque service.

6. **Vecteur de description de service de client** : Soit $R = \{R_1, R_2, R_3, \dots, R_m\}$ c'est l'ensemble de QoS entré par le client, tel que R_i est la valeur minimale des critères. La sélection de service se fait par la comparaison entre les vecteurs D et R afin de choisir les meilleurs services.

7. **Vecteur de poids** : Soit $W = \{W_1, W_2, W_3, \dots, W_m\}$ représente un vecteur de poids des critères qui prend la valeur 1 par défaut. Chaque client peut exprimer la propriété de chaque critère en attribuant des poids pour chaque critère.

La sélection de service implique une comparaison entre le vecteur de critères d'exigence de client et tous les vecteurs de fournisseur de service, puis la sélection du service dont le vecteur de fournisseur correspondant le mieux au vecteur d'exigence de client. Il s'agit essentiellement d'une mesure de similarité entre le vecteur d'exigence de client et les vecteurs de fournisseurs de service.

Deux approches différentes pour mesurer la similarité ont été proposées :

- Dans la première méthode, ils ont appelé la différence pondérée WD , ils soustraient le vecteur d'exigence de l'utilisateur de chaque ligne de la matrice de décision pour obtenir :

$$\begin{pmatrix} a_{1,1} - r_1 & a_{1,2} - r_2 & \cdot & \cdot & \cdot & a_{1,n} - r_n \\ a_{2,1} - r_1 & a_{2,2} - r_2 & \cdot & \cdot & \cdot & a_{2,n} - r_n \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{l,1} - r_1 & a_{l,2} - r_2 & \cdot & \cdot & \cdot & a_{l,n} - r_n \end{pmatrix}$$

- Puis, ils ont calculent le produit entre le vecteur de poids W et résultat de la soustraction de la matrice M qui donne ce vecteur de colonne :

$$\begin{pmatrix} (a_{1,1} - r_1)w_1 & (a_{1,2} - r_2)w_2 & \cdot & \cdot & \cdot & (a_{1,n} - r_n)w_n \\ (a_{2,1} - r_1)w_1 & (a_{2,2} - r_2)w_2 & \cdot & \cdot & \cdot & (a_{2,n} - r_n)w_n \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ (a_{l,1} - r_1)w_1 & (a_{l,2} - r_2)w_2 & \cdot & \cdot & \cdot & (a_{l,n} - r_n)w_n \end{pmatrix}$$

- Ainsi, ils ont multiplions la matrice par le scalaire (-1) , puis remplacent chaque élément par e élevé à la puissance de l'élément respectif lui-même.
- Ensuite, ils ont calculé la somme de chaque ligne de la matrice et les placent dans le vecteur dans un vecteur ewD .

$$\begin{pmatrix} e^{-(a_{1,1}-r_1)w_1} + e^{-(a_{1,2}-r_2)w_2} + \cdot & \cdot & \cdot & +e^{-(a_{1,n}-r_n)w_n} \\ e^{-(a_{2,1}-r_1)w_1} + e^{-(a_{2,2}-r_2)w_2} + \cdot & \cdot & \cdot & +e^{-(a_{2,n}-r_n)w_n} \\ \cdot & \cdot & \cdot & \cdot \\ e^{-(a_{l,1}-r_1)w_1} + e^{-(a_{l,2}-r_2)w_2} + \cdot & \cdot & \cdot & +e^{-(a_{l,n}-r_n)w_n} \end{pmatrix}$$

- Enfin ,chaque élément du vecteur EWD résultant est une évaluation de conformité de service correspondant aux exigences de l'utilisateur et l'élément ayant une valeur minimale correspond au service le plus approprié.

II.5 Composition des services Cloud

Les services Cloud sont atomique et chaque service assure l'exécution d'une seule fonctionnalité, alors que la requête de l'utilisateur est parfois composée de nombreuses fonctionnalités, c'est pourquoi un service ne peut pas satisfaire la demande de l'utilisateur, mais il faut plusieurs services pour donner à l'utilisateur le résultat attendu, il doit communiquer et être combiné avec d'autres services. Dans ce cas, il s'agit de la composition des services ou « service composite ».

II.5.1 Définition

Selon [42], la composition est un processus qui permet de générer des services complexes à partir de services atomiques. Ces derniers ont la faculté de négocier et interagir de façon intelligente afin de découvrir automatiquement d'autres services qui servent à l'enrichissement du service composite.

Selon [43], la composition est un mécanisme qui permet de regrouper un ensemble de services, qui doivent satisfaire une requête complexe, en utilisant des structures de contrôle et d'échange de messages.

En gros, La composition des services est la capacité de combiner deux ou plusieurs services existants pour créer de nouveaux services en combinant les fonctions des services candidats pour répondre aux besoins spécifiques des utilisateurs.

II.5.2 Catégorisation des mécanismes de composition

Selon [43], les mécanismes de composition peuvent être classés en 4 catégories :

- **Composition Proactive** : La composition proactive est également appelée composition statique ou composition hors ligne, dans ce type de composition, les services à combiner doivent être présélectionnés et liés de manière fixe. C'est à dire qu'ils ne peuvent pas être modifiés. La composition proactive ne répond pas nécessairement aux

exigences des utilisateurs, puisqu'elle est effectuée avant l'arrivée de leurs requêtes.

- **Composition Réactive :** La composition réactive est également appelée composition dynamique ou composition en-ligne, ce qui fait référence à la sélection en temps réel et à la combinaison de services en fonction des besoins des utilisateurs. Une fois que la demande de l'utilisateur arrive, la sélection des services et la liaison de ces services entre eux sont effectuées pour générer un nouveau service.
- **Composition Obligatoire :** Tous les services atomiques composant doivent obligatoirement participer à l'exécution du service composite.
- **Composition Optionnel :** La participation d'un certain nombre de services atomiques composants est suffisante pour la réussite de l'exécution du service composite.

II.6 Les problèmes de la découverte des services Cloud

Les problèmes de la découverte et de la sélection des services Cloud sont [2], [3] :

- **Manque de standardisation :**

Le manque de standardisation représente le problème principal de la technologie de Cloud Computing. Chaque fournisseur de Cloud utilise son vocabulaire particulier pour présenter ses services sur Internet, où la majorité des fournisseurs dépendent du langage naturel pour présenter leurs services. L'absence d'une spécification de service Cloud standardisée rend les processus de découverte et de sélection plus complexe pour les utilisateurs, en particulier ceux qui ont une faible expérience des services Cloud.

- **Portabilité et interopérabilité :**

La portabilité est la capacité de déplacer des données entre différents fournisseurs Cloud, tandis que l'interopérabilité est définie comme étant la facilité de migration des données. À cause de verrouillage des fournisseurs, la conversion des données du format du système source au format requis par le système cible peut demander beaucoup d'efforts car les fournisseurs de services Cloud Computing n'utilisent pas tous les mêmes modèles et technologies pour les descriptions de ses données. En cas de sélection inappropriée d'un service Cloud, un client Cloud peut être confronté au problème de la dépendance vis-à-vis du fournisseur et à des problèmes de portabilité et d'interopérabilité des données.

- **Manque d'ontologie de service Cloud standardisée complète :**

Les études de découverte existantes visent à résoudre le problème de la normalisation

et à fournir un modèle de spécification homogène pour décrire les services Cloud en utilisant les technologies sémantiques incarnées dans l'utilisation massive de l'ontologie. L'ontologie Cloud peut capturer et masquer l'hétérogénéité et la diversité des FF et des NFF en catégorisant les services existants en groupes et en identifiant leurs caractéristiques communes. L'ontologie Cloud peut fournir une spécification unifiée et explicite des services Cloud. Cela peut contribuer à faciliter le processus de découverte pour les utilisateurs du Cloud et, à son tour, à accélérer et à améliorer l'efficacité de la récupération.

- **Connaissance insuffisante du concept de Cloud Computing :**

Les utilisateurs préfèrent utiliser leurs mots naturels pour composer leurs requêtes plutôt que d'utiliser des requêtes basées sur des mots-clés. La composition de requêtes textuelles à partir des concepts disponibles de l'ontologie du Cloud référencée est un obstacle pour les utilisateurs ayant une faible expérience ou une faible connaissance des termes et concepts du Cloud (les utilisateurs non experts). Ainsi, de nombreuses entreprises et utilisateurs abandonnent les services Cloud car ils n'ont pas suffisamment de connaissances sur les concepts du Cloud.

- **Grand nombre de services Cloud publiés :**

Le nombre de services fournis augmente chaque jour, ce qui rend aux utilisateurs un problème difficile de découvrir les services et de choisir le meilleur service et le service le plus adapté en fonction de ses propres besoins.

II.7 Conclusion

Les chercheurs ont joué un rôle important et efficace dans le développement rapide et étonnant du Cloud Computing. Ils ont introduit de nombreuses fonctions et outils dans le domaine de la description et de la découverte de services Cloud.

Dans ce chapitre, nous avons présenté les dernières technologies concernant la description des services Cloud et les concepts de découverte, de sélection et de composition.

Dans le prochain chapitre, nous allons survoler quelques travaux connexes dans ce domaine.

Chapitre III

Solutions existantes pour la découverte des services dans le Cloud

III.1 Introduction

Les problèmes mentionnés dans le chapitre précédent ont fait l'objet de plusieurs efforts ayant visé la résolution de la problématique de découverte. Dans ce chapitre nous présenterons ces efforts. Nous discuterons les solutions proposées pour réaliser la découverte dans le Cloud Computing.

III.2 Travaux connexes

Dans cette partie, nous mentionnerons certains travaux précédents dans la description et la découverte des services de Cloud Computing.

III.2.1 Solution de Rekik et al [44]

Cette solution se concentre sur la structuration des services de Cloud Computing sous la forme d'une ontologie afin de satisfaire la découverte des services Cloud. Pour résoudre le problème de l'hétérogénéité de description des services Cloud. Les auteurs ont proposé dans [44] une approche basée sur l'ontologie pour la découverte et la sélection de service. Ils ont proposé une ontologie qui englobe toutes les catégories des services Cloud (IaaS, PaaS et SaaS) et qui est basée sur plusieurs caractéristiques de plusieurs standards. De plus, cette ontologie prend en compte les propriétés fonctionnelles et non fonctionnelles. Pour bien com-

prendre les notions traitées dans ce travail nous avons réalisé une ontologie en utilisant l'outil Protégé (voir Annexe A).

Cette solution est basée sur la description des services Cloud, elle a une ontologie bien structurée, classifiée en trois modèles : IaaS, PaaS et SaaS, elle prend en charge les fonctionnalités fonctionnelles et non fonctionnelles de chaque service ainsi qu'elle vérifie l'aspect opérationnel qui décrit l'interface et les fonctionnalités des services. Mais elle ne prend pas en charge l'aspect commercial qui décrit les acteurs qui interagissent avec un service comme le SLA, ainsi que l'aspect technique qui décrit les technologies utilisées par le service comme les points de liaisons, etc. Aussi, elle se focalise seulement sur la description des services et non pas sur la découverte et la sélection des services Cloud.

III.2.2 Solution de Al-Sayed et al [2]

Les auteurs de cette solution ont proposé tout un framework. Ce framework contient 5 couches principales (Couche utilisateur, Métadonnées, BDR, accès aux données et Découverte). Pour bien comprendre les différentes couches traitées dans cette solution nous avons réalisé une application en utilisant l'environnement JavaEE pour la description et la découverte des services et nous avons implémenté les différentes couches (voir Annexe A).

Dans cette solution les services Cloud sont traités comme des produits et enregistrés dans un référentiel Cloud unifié dédié pour tout les fournisseurs de services, où les services sont classés en fonction de leurs FF provisionnés dans les concepts de l'ontologie Cloud développée en tenant compte des NFF de chaque catégorie.

Parmi les avantages de cette solution on peut citer :

Cette solution permet aux utilisateurs ayant une faible connaissance des concepts du Cloud de composer des requêtes complexes en utilisant leur langage naturel, elle permet aussi aux fournisseurs de services Cloud d'utiliser leur propre vocabulaire pour présenter leurs services. Par conséquent cette solution prend en considération le traitement de requête d'utilisateur et de publication de fournisseur. Ainsi que l'utilisation d'une ontologie pour unifier la description des services enregistrés par les fournisseurs et pour fournir un modèle de représentation sémantique des données dans plusieurs domaines Cloud et l'utilisation d'une BDR pour le stockage des données. De plus, l'utilisation de Matching donne un résultat exact qui comprend une liste des FF correspondant à la requête d'utilisateur. Elle permet de donner la

main à l'administrateur informatique pour maintenir les données à jour.

En ce qui concerne les inconvénients, on peut citer :

Lors de la recherche d'un service, la recherche passera par tous les services de tous les domaines au lieu de simplement rechercher dans le domaine de service demandé. Dans cette solution, les auteurs ont supposé que la BD est entièrement remplie de services avant l'arrivée du premier fournisseur. Si un fournisseur ajoute une FF qui n'existait pas auparavant, cela signifie que son domaine n'est pas disponible dans l'ontologie. Dans ce cas, les services n'est pas enregistré dans la BD. Nous remarquons aussi que malgré l'utilisation de Ranking la sélection avec QoS n'est pas prise en compte et le Ranking des services donne un mauvais résultat. De plus il y'a une erreur de calcul dans le cas ou le Matching retourne une seule FF le Ranking divise sur le $\text{Log}(1)$ ce qui donne une infinité. Cette solution impose au fournisseur de saisir ses services à travers une interface au lieu d'utiliser des API.

III.2.3 Solution de Abbas et al [3]

L'architecture de cette solution est basée sur un SMA. Le SMA est un système constitué d'un ensemble d'agents pour différentes tâches dans le domaine du Cloud. Pour bien comprendre les concepts abordés dans ce travail, nous avons implémenté une ontologie en utilisant l'outil Protégé, nous avons créé une application en utilisant l'environnement JavaEE, ainsi qu'un SMA en utilisant le framework JADE pour la découverte et la sélection des services (voir Annexe A).

Cette solution répond aux problèmes liés à l'interopérabilité du Cloud Computing, et s'appuie sur les critères de qualité de service (QoS) pour effectuer les traitements de sélection afin de répondre à la requête des utilisateurs.

Parmi les avantages de cette solution on peut citer :

L'utilisation des agents facilite les tâches, chaque agent à une tache spécifique à résoudre et l'indépendance des agents facilite la modification de leur comportements. Ainsi, le temps d'exécution est rapide par rapport aux autres solutions qui n'utilisent pas des agents. De plus, cette solution donne au client la possibilité d'exprimer son point de vue sur le service dont il a bénéficié. En plus l'utilisation d'une ontologie pour unifier la description des services enregistrés par les fournisseurs et pour fournir un modèle de représentation sémantique des services.

Et pour les inconvénients que l'on peut citer :

La communication entre le Broker et le fournisseur se fait via des agents s'il n'y a pas d'agents, il n'y aura pas de communication, donc cette solution oblige le fournisseur à avoir un agent à son niveau au lieu d'utiliser des API, elle impose aussi au fournisseur et au consommateur de saisir leur requête de recherche à l'aide de mots clés au lieu d'utiliser le langage naturel, donc cette solution ne prend pas en compte le traitement de requête des utilisateurs et de publication des fournisseurs, nous pouvons ajouter que l'ontologie n'est pas détaillée et ne vérifie pas l'aspect technique et commercial de plus les services ne sont pas classifiés en domaines cela rend le processus de découverte plus long, donc lors de la recherche d'un service la recherche passera par tous les services de tous les domaines au lieu de rechercher seulement sur le domaine de service recherché, ainsi que le stockage des services se fait dans l'ontologie tandis que l'objectif principal des ontologies est de fournir un modèle de représentation sémantique des données avec un accent minimal sur la structure des instances de stockage.

III.2.4 Solution de Kang et al [45]

Cette solution présente un système de découverte de service Cloud basé sur un SMA. Ils ont utilisé une ontologie pour donner aux services une représentation sémantique et pour faire le raisonnement entre les services demandés et publiés, tout ça pour sélectionner le service optimal selon la demande de l'utilisateur.

Afin de mieux comprendre les notions traitées dans la solution [45] nous avons réalisé une ontologie à l'aide de l'outil Protégé ainsi une application à l'aide de l'environnement JavaEE pour la découverte des services et nous avons implémenté un SMA à l'aide de framework JADE (voir Annexe A).

Parmi les avantages de cette solution on peut citer :

Cette solution répond au problème de saturation du système (Broker) avec de nombreuses demandes, un mécanisme dépendant de plusieurs agents Broker étant proposé pour faire face au problème d'évolutivité. Par exemple, avec plusieurs agents Brokers, même si la connexion entre les agents utilisateurs et les agents fournisseurs a échoué dans un agent Broker, l'agent Broker peut envoyer le message de recommandation à d'autres agents Brokers à l'aide d'une base de données qui stocke et garde une trace des données historiques pour chaque agent

afin de faire une recommandation intelligente. Ils ont proposé un système de caches (files d'attente) pour stocker les demandes des clients et les publications des fournisseurs aux agents Broker, et si l'agent est saturé, il envoie les demandes et les publications à un autre agent pour équilibrer la charge. Ainsi que l'utilisation d'une ontologie pour unifier la description des services enregistrés par les fournisseurs et pour fournir un modèle de représentation sémantique pour les services.

Et pour les inconvénients que l'on peut citer :

La probabilité que la publication demandée par l'utilisateur existe dans la file d'attente dans le premier agent est très faible ce qui nécessite un passage aux autres agents sachant que chaque agent communique avec tous les agents du système ce qui génère un temps d'exécution très élevé, et aussi un problème de complexité parce que cette solution dépend du nombre de services et aussi du nombre d'agents, plus le nombre de services et le nombre d'agents sont élevés, plus la complexité est élevée. Les files d'attente sont des systèmes de caches qui ont un espace de stockage très petit, donc on ne peut pas compter seulement sur ces files d'attente, car si le système tombe en panne, toutes les files d'attente seront vidées, de plus dans cette solution, il n'y a pas de système pour vider ces files d'attente, une fois les files d'attente saturées, elles le restent jusqu'à ce que le système tombe en panne. Il faut donc stocker les services dans des entrepôts distincts. Aussi, ils n'ont pas mentionné plus de détails concernant le calcul de similarité dans la phase de recommandation. De plus, ils ont utilisé une description simple des services qui nécessitent un aspect technique et commercial de plus les services ne sont pas classifiés en domaines cela rend le processus de découverte plus long, donc lors de la recherche d'un service la recherche va passer par tous les services de tous les domaines au lieu de faire la recherche seulement sur le domaine de service recherché. Nous pouvons ajouter que cette solution impose aux utilisateurs et aux fournisseurs de saisir leur requête de recherche et de publication respectivement en utilisant des mots-clés au lieu d'utiliser le langage naturel donc ils n'ont pas considéré en compte le traitement de la requête. Elle exige aussi au fournisseur d'avoir un agent à son niveau plutôt que d'utiliser des API.

III.2.5 Solution de Kwang [46]

Cette solution présente une nouvelle architecture de Cloudle composée d'un agent de découverte de services (SDA), d'une ontologie Cloud, d'une base de données de services Cloud, de

plusieurs crawlers Cloud et d'une interface Web (voir Annexe A).

Lors de la construction d'un moteur de recherche général (par exemple Google), il suffit de considérer la question de la recherche de pages Web qui contiennent des concepts dans la requête d'un utilisateur. Le problème de la construction d'un moteur de recherche de services Cloud est plus complexe car il faut rechercher des services répondant à trois types d'exigences (les exigences fonctionnelles, techniques et budgétaires). Le moteur de recherche dans ce travail emploie un agent de découverte de services qui consulte une ontologie Cloud pour déterminer les similitudes entre les spécifications fonctionnelles et techniques des services des fournisseurs et les exigences fonctionnelles et techniques des consommateurs.

Parmi les avantages de cette solution on peut citer :

L'utilisation d'une ontologie pour unifier la description des services enregistrés par les fournisseurs et pour fournir un modèle de représentation sémantique des données et l'utilisation d'une BDR pour le stockage des données. De plus ils ont proposé une découverte par crawlers pour collecter les services de manière automatique à partir de www.

En ce qui concerne les inconvénients, on peut citer :

Cette solution a une ontologie détaillée mais les services ne sont pas classés en domaines et cela rend le processus de découverte plus long, Parce que tous les domaines qu'ils ont traités existent dans une seule ontologie. Donc, lors de la recherche d'un service, la recherche passera par tous les services de tous les domaines au lieu de simplement rechercher dans le domaine de service demandé. Nous pouvons ajouter que cette solution impose aux utilisateurs et aux fournisseurs de saisir leur requête de recherche et de publication respectivement en utilisant des mots-clés au lieu d'utiliser le langage naturel donc ils n'ont pas considéré en compte le traitement de la requête. Ainsi que cette solution impose au fournisseur de saisir ses services à travers une interface au lieu d'utiliser des API.

III.2.6 Solution de Han et al [47]

Selon [47]. Quand un utilisateur recherche un service par un moteur de recherche, il obtient un nombre énorme de résultats, ainsi que le moteur de recherche nous retourne que des liens URL contenant des pages Web non pertinentes, donc il y a un problème de qualité de service, ainsi, l'utilisateur perd beaucoup de temps pour visiter tous les URL, et dans le résultat retourné il n'existe aucune relation entre les services parce que chaque service à un format

différent et chaque service existe dans un entrepôt différent, donc le moteur de recherche ne peut pas choisir le meilleur service. Alors il faut faire un mécanisme pour découvrir les services de Cloud et relation entre les services. Taekgyeong Han et Kwang Mong Sim ont mis en place un système (CSDS) pour découvrir et trouver le meilleur service Cloud (voir Annexe A).

Cet article présente un système de découverte de services Cloud. Il est spécialement conçu pour les utilisateurs qui souhaitent trouver un Cloud service sur Internet. Cette solution est centrée sur le fournisseur plus que l'utilisateur.

Pour les avantages de cette solution on peut citer :

Cette solution donne la main au utilisateur de passer sa recherche tout seul sans aucune intervention en utilisant un moteur de recherche, elle facilite la tâche aux fournisseurs, d'habitude la recherche d'un service Cloud se fait par rapport aux services qui sont enregistré dans un entrepôt particulier mais cette solution est ouverte donc on peut dire que l'architecture de cette solution est décentralisée. Elle a donné l'idée d'une recherche d'informations, elle n'a pas restreint la recherche juste aux services qui sont au niveau de l'entrepôt de Broker, mais sa mise en œuvre actuellement n'est pas évidente (par exemple en vas passer avec le recensement de tous les fournisseurs et ce n'est pas évident).

En ce qui concerne les inconvénients, on peut citer :

L'article de cette solution n'a pas donné trop de détail concernant le fonctionnement de son système il été très limité, temps de réponse est lent à cause de la recherche des services sur le net, chaque fournisseur utilise son propre format et chaque service existe dans un entrepôt différent. Cette solution ne gère pas la communication entre le fournisseur et le client, elle donne seulement le résultat au client sous forme des pages web rien de plus. Donc dans le cas d'un client non expert il va trouver des difficultés pour choisir le meilleur service qui lui convient. Les auteurs de cette solution ont simulé l'implémentation du travail, ils n'ont pas travaillé un travail réel lors de la simulation ils ont proposé leurs propres pages (page virtuelle des fournisseurs) pour faciliter les tâches. Ce n'est pas facile d'implémenter cette solution car on ne connaît pas tous les fournisseurs, leurs ressources et leurs emplacements.

III.2.7 Solution de Hasan et al [48]

Cet article présente une ontologie de domaine pour les services Cloud afin de créer une compréhension partagée de l'environnement Cloud et de modéliser les services Cloud réels. À la connaissance des auteurs, il s'agit de la première tentative de création d'un marché des services Cloud où les fournisseurs de Cloud et les consommateurs de Cloud peuvent transformer les services Cloud en services publics. Le cadre de cette solution est divisé en six sous-systèmes et dix composants (voir Annexe A).

Cet article a présenté un cadre générique pour les services Cloud dans le marché basé sur une ontologie Cloud unifiée. De plus, cet article a mis en œuvre deux instances de ce cadre, l'une est basée sur l'approche des attributs dominants et récessifs et l'autre est basée sur l'approche de similarité sémantique ontologique.

Parmi les avantages de cette solution on peut citer :

Cette solution permet à l'utilisateur de choisir entre saisir sa requête en langage naturel ou utilise des listes prédéfinies, des cases à cocher ou des boutons radio pour saisir des requêtes donc elle prend en considération le traitement de la requête d'utilisateur. Ainsi que l'utilisation d'une ontologie pour unifier la description des services enregistrés par les fournisseurs et pour fournir un modèle de représentation sémantique des données et l'utilisation d'une BDR pour le stockage des données. En plus, l'utilisation de Matching pour rechercher des services similaires à la demande de l'utilisateur. En plus de donner au client la possibilité d'exprimer son point de vue sur le service dont il a bénéficié. De plus, l'utilisation d'un système de surveillance pour vérifier que le service Cloud respecte l'accord de niveau de service (SLA).

En ce qui concerne les inconvénients, on peut citer :

L'article de cette solution n'a pas donné trop de détail concernant le fonctionnement de son système par exemple comment calculer le Matching. Les services de l'ontologie de cette solution ne sont pas classifiés en domaines cela rend le processus de découverte plus long, car tous les domaines qu'ils ont traités existent dans une seule ontologie, donc lors de la recherche d'un service la recherche va passer par tous les services de tous les domaines au lieu de faire la recherche seulement sur le domaine de service recherché. Cette solution impose au fournisseur de saisir ses services à travers une interface au lieu d'utiliser des API.

III.2.8 Solution de Beheshti et al [49]

Cette solution présente un moteur de recherche de services Cloud (CSSE). Avec l'adoption croissante du Cloud Computing, pour trouver le Cloud pertinent pour les clients devient effectivement un enjeu de recherche important. Cet article, a mené une analyse complète des services Cloud actuellement disponible sur le Web. Ils ont développé un moteur de recherche de services Cloud qui collecte, extrait et identifie les services Cloud. Ces résultats offrent une vue d'ensemble sur l'état actuel des services Cloud sur le World Wide Web (voir Annexe A). Parmi les avantages de cette solution on peut citer :

L'utilisation de clustering pour collecter les services Cloud les plus similaires dans des clusters, afin de gagner du temps d'exécution.

En ce qui concerne les inconvénients, on peut citer :

Cet article de solution n'a pas fourni beaucoup de détails sur le fonctionnement de son système, il était très limité. Donc ce n'est pas facile de comprendre cette solution

III.3 Comparaison des différents travaux

Le tableau ci-dessous présente un résumé des différentes solutions que nous avons examinées et que nous avons présentées dans les sections précédentes, selon les critères suivants :

- **C1 «Aspect Technique»** : Décrivent les besoins en termes de technologies utilisés par le service comme l'emplacement de service, les protocoles qu'il utilise, les points de liaisons, etc.
- **C2 «Aspect Sémantique»** : Se distingue par l'utilisation d'ontologie.
- **C3 «Aspect Opérationnel»** : Décrivent l'interface et la fonctionnalité des services en détails comme les fonctionnalités offertes par le service, URL de service, etc.
- **C4 «Aspect Commercial»** : Décrit les acteurs qui interagissent avec un service, les SLA, etc.
- **C5 «Requête Utilisateur»** : Le format de la requête saisit par l'utilisateur pour découvrir des services.
- **C6 «Requête Fournisseur»** : Le format de la requête saisit par le fournisseur pour découvrir des fonctionnalités.
- **C7 «Domaine Couvert»** : Est ce que la solution garantit la description, la découverte et la sélection.

- **C8 «Description»** : Est ce que l'ontologie est bien détaillée.
- **C9 «Temps de réponse»** : Est ce que l'exécution de la solution prend beaucoup de temps.
- **C10 «Classification des services»** : Est ce que l'ontologie a classifié les services.
- **C11 «Stockage des Services»** : Où sont stockés les services.
- **C12 «Service de déploiement»** : Les services couverts par l'ontologie.
- **C13 «Architecture»** : Si la solution est un composant de traitement centralisée ou décentralisée.
- **C14 «Ontologies utilisées»** : Si la solution contient une ontologie de domaine et/ou de description.

<i>Solution</i> <i>Critre</i>	Solution 01	Solution 02	Solution 03	Solution 04
C1	Non traité	Non traité	Non traité	Non traité
C2	Traité	Traité	Traité	Traité
C3	Traité	Traité	Traité	Traité
C4	Non traité	Traité	Non traité	Non traité
C5	/	Langage naturel	Mots-clés	Mots-clés
C6	/	Langage naturel	Mots-clés	Mots-clés
C7	Description	Description - Découverte	Description - Découverte - Sélection	Description - Découverte - Sélection
C8	Détaillée	Détaillée	Non-Détaillée	Non-Détaillée
C9	/	Lent	Rapide	Lent
C10	Traité	Non traité	Traité	Traité
C11	Ontologie	BDR	Ontologie	Système de cache
C12	IaaS, PaaS et SaaS	IaaS, PaaS et SaaS	IaaS, PaaS et SaaS	IaaS, PaaS, SaaS, DaaS et CaaS
C13	/	Centralisée + OBDA	Centralisée + SMA	Centralisée + SMA
C14	Description	Domaine + Description	Description	Description

<i>Solution</i>	Solution 05	Solution 06	Solution 07	Solution 08
<i>Critre</i>				
C1	Non traité	Non traité	Non traité	Non traité
C2	Traité	Traité	Traité	Traité
C3	Traité	Traité	Traité	Traité
C4	Non traité	Non traité	Traité	Non traité
C5	Mots clés	Mots clés	Langage naturel - Mots clés	Mots clés
C6	Mots clés	Mots clés	Langage naturel - Mots clés	Mots clés
C7	Description - Découverte	Description - Découverte	Description - Découverte	Découverte
C8	Détaillée	Non-Détaillée	Détaillée	Non-Détaillée
C9	Lent	Lent	Lent	Rapide
C10	Traité	Traité	Traité	Traité
C11	BD	Ontologie	BD	BD
C12	IaaS, PaaS, SaaS, DaaS et CaaS	IaaS, PaaS, SaaS	IaaS, PaaS, SaaS	IaaS, PaaS, SaaS
C13	Centralisée + SMA	Décentralisée + SMA	Centralisée	Centralisée
C14	Description	Description	Description	/

TAB. III.1 : Comparaison entre les différentes solutions

III.4 Discussion

Notre étude comparative des solutions nous a permis de construire le tableau (III.1), et nous avons constaté que la plupart des solutions sont axées sur le niveau PaaS, SaaS et IaaS.

Par manque de standardisation de description, les travaux étudiés ont choisi de décrire les services de déploiement traités dans leurs solutions à travers leurs propres modèles. Ils étaient basés sur les ontologies, ceci leur a permis de prendre en charge l'aspect sémantique dans la description de leurs solutions.

Certains des travaux ont décrit les services en considérant certains aspects. Nous trouvons des travaux qui traitent l'aspect commercial comme la solution "[2], [48]" ainsi que l'aspect sémantique et opérationnel qui sont considérés dans tous les travaux. Nous avons aussi des travaux qui incluent dans leurs solutions la description comme "[2], [3], [44], [45], [46], [47], [48]", la découverte comme "[2], [3], [45], [46], [47], [48], [49]" et la sélection comme "[3], [45]". Ainsi que la requête de l'utilisateur et de fournisseur qui varie d'une solution à l'autre (langage naturel/mots clés). Sans oublier le stockage des services, nous avons des travaux qui stockent leurs services dans une BDR comme la solution "[2], [46], [48], [49]" et des travaux qui stockent leurs services dans une ontologie comme la solution "[44], [3], [47]". Cela signifie que de nombreuses approches différentes et solutions possibles ont été publiées, mais il n'existe pas encore de standard accepté.

III.5 Conclusion

Dans ce chapitre nous avons présenté des travaux proposés dans le contexte de la découverte de service pour avoir une idée sur les différentes solutions proposées. Nous avons terminé le chapitre par une étude comparative des différentes approches proposées, tenant compte de plusieurs critères, pour voir la différence entre ces approches et pour définir les objectifs de recherche pour chaque approche.

Dans le prochain chapitre, nous allons présenter la modélisation de notre solution proposée, ce chapitre expliquera en détail les étapes de description, de découverte et de sélection.

Chapitre IV

Conception de la solution

IV.1 Introduction

Dans le chapitre précédant nous avons comparé les différentes solutions proposées dans la littérature. Cette étude nous a facilité la réalisation de notre solution qui va être présentée dans ce chapitre.

Dans ce chapitre, nous allons commencer par parler de notre démarche de travail. Puis, nous aurons une section dédiée à l'analyse des besoins où nous détaillons les acteurs et chacune de leurs interactions avec le système. Ensuite, nous allons présenter notre solution proposée en montrant l'architecture de notre système et en détaillant chaque fonctionnalité essentielle à l'aide de pseudo-algorithmes et d'exemples explicatifs.

IV.2 Démarche de travail

Tout au long des différentes phases de réalisation de notre projet, nous avons adopté la méthode eXtreme Programming (XP)¹, il s'agit d'une méthode agile plus particulièrement orientée sur l'aspect réalisation d'une application, sans pour autant négliger l'aspect gestion de projet. Cette méthode est conçue pour améliorer la qualité du logiciel et sa capacité à s'adapter correctement aux besoins changeants du client.

La méthode est définie comme :

- Une tentative de réconcilier l'humain avec la productivité.
- Un mécanisme pour faciliter le changement social.

1. <http://www.extremeprogramming.org/>

- Une voie d'amélioration.
- Un style de développement.
- Une discipline de développement d'applications informatiques.

Son but principal est de réduire les coûts du changement. Dans les méthodes traditionnelles, les besoins sont définis et souvent fixés au départ du projet informatique ce qui accroît les coûts ultérieurs de modifications. XP s'attache à rendre le projet plus flexible et ouvert au changement en introduisant des valeurs de base, des principes et des pratiques.

L'extreme programming repose sur cinq valeurs fondamentales :

1. **Communication** : C'est le moyen fondamental pour éviter les problèmes. Les tests, la programmation en binôme et le jeu du planning obligent les développeurs, les décideurs et les clients à communiquer. Si un manque apparaît malgré tout, un coach se charge de l'identifier et de remettre ces personnes en contact.
2. **Simplicité** : La façon la plus simple d'arriver au résultat est la meilleure. Une application simple sera plus facile à faire évoluer.
3. **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.
4. **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.
5. **Respect** : Cette valeur inclut le respect pour les autres, ainsi que le respect de soi. Les programmeurs ne devraient jamais valider les modifications qui cassent la compilation, qui font échouer les tests unitaires existants ou qui retardent le travail de leurs pairs. Les membres respectent leur propre travail en cherchant toujours la qualité et la meilleure conception pour la solution.

IV.3 Spécification des besoins

Pour spécifier les besoins du système il faut tout d'abord spécifier les acteurs qui interagissent avec le système, qui sont :

- **Fournisseur** : L'acteur qui fournit les services Cloud, il doit s'inscrire dans le système afin de pouvoir s'identifier et publier des services, consulter et mettre à jour la liste des services qu'il a publiés.
- **Utilisateur** : L'acteur qui désire profiter du service, il doit s'inscrire dans le système pour pouvoir demander et consommer un service, de même il peut évaluer le service dont il a bénéficié.

Le diagramme de cas d'utilisation suivant (IV.1) montre les actions possibles entre les acteurs et le système :

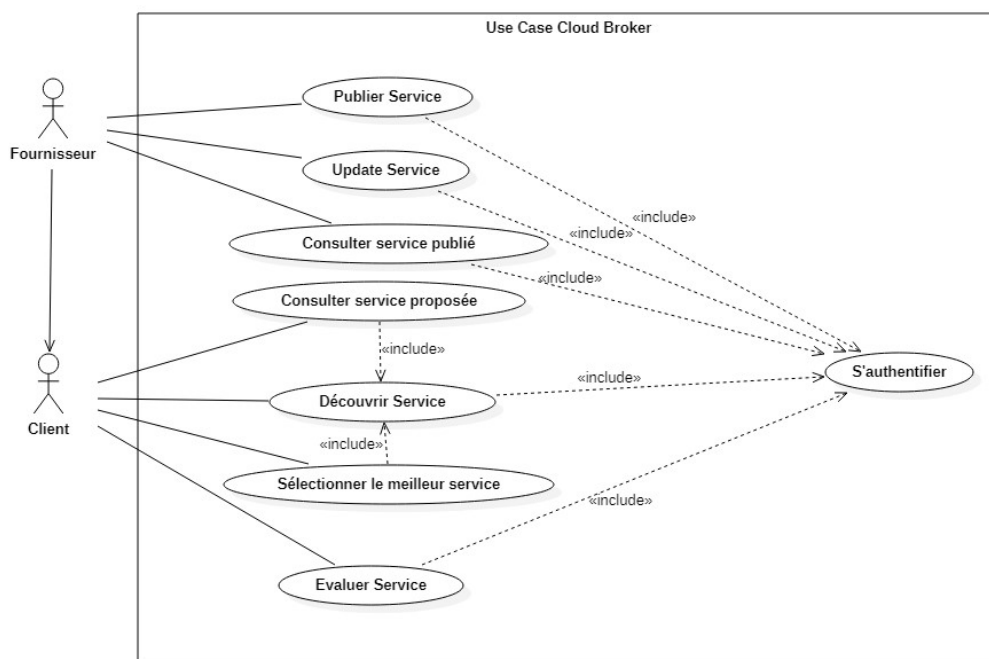


FIG. IV.1 : Diagramme de cas d'utilisation "Cloud Broker"

IV.4 Solution Proposée

Dans cette section nous présenterons la modélisation de notre solution. Pour rappel, notre système est un courtier de Cloud Computing pour la découverte des services Cloud.

Pour faire une bonne découverte, nous avons un ensemble de règles qui doivent être vérifiés dans notre solution, nous pouvons citer :

Règle 1: Une bonne découverte doit se faire en temps d'exécution rapide.

Règle 2: Pour diminuer le temps de recherche il faut classer les services en clusters de domaines.

Règle 3: Une description détaillée et standard des services facilite toutes les opérations sur les services.

Règle 4: Une bonne description prend en charge l'aspect fonctionnel et non fonctionnel des services.

Règle 5: Utilisation d'une ontologie pour unifier la description des services enregistrés par les fournisseurs et pour fournir un modèle de représentation sémantique des données en plusieurs domaines Cloud pour faciliter la découverte.

Règle 6: Pour une bonne découverte l'aspect sémantique ne doit pas être négliger.

Règle 7: Une bonne découverte donne une bonne sélection.

Règle 8: Pour une bonne sélection il faut prendre en considération les critères de qualité de service.

Règle 9: Pour faciliter la découverte au client il faut proposer une recherche par langage naturel.

Règle 10: Le fournisseur doit publier son service via une API.

La découverte a déjà été présentée dans le chapitre II, mais nous rappelons brièvement qu'il existe plusieurs services déployés dans le Cloud avec différents formats, il est alors difficile de trouver un service approprié à notre besoin avec une bonne valeur de qualité de service. L'architecture suivante (IV.2) montre les éléments de notre système. Cette architecture est basée sur un système multi-agents inspiré de la solution [3] que nous avons déjà implémenté (voir Annexe A).

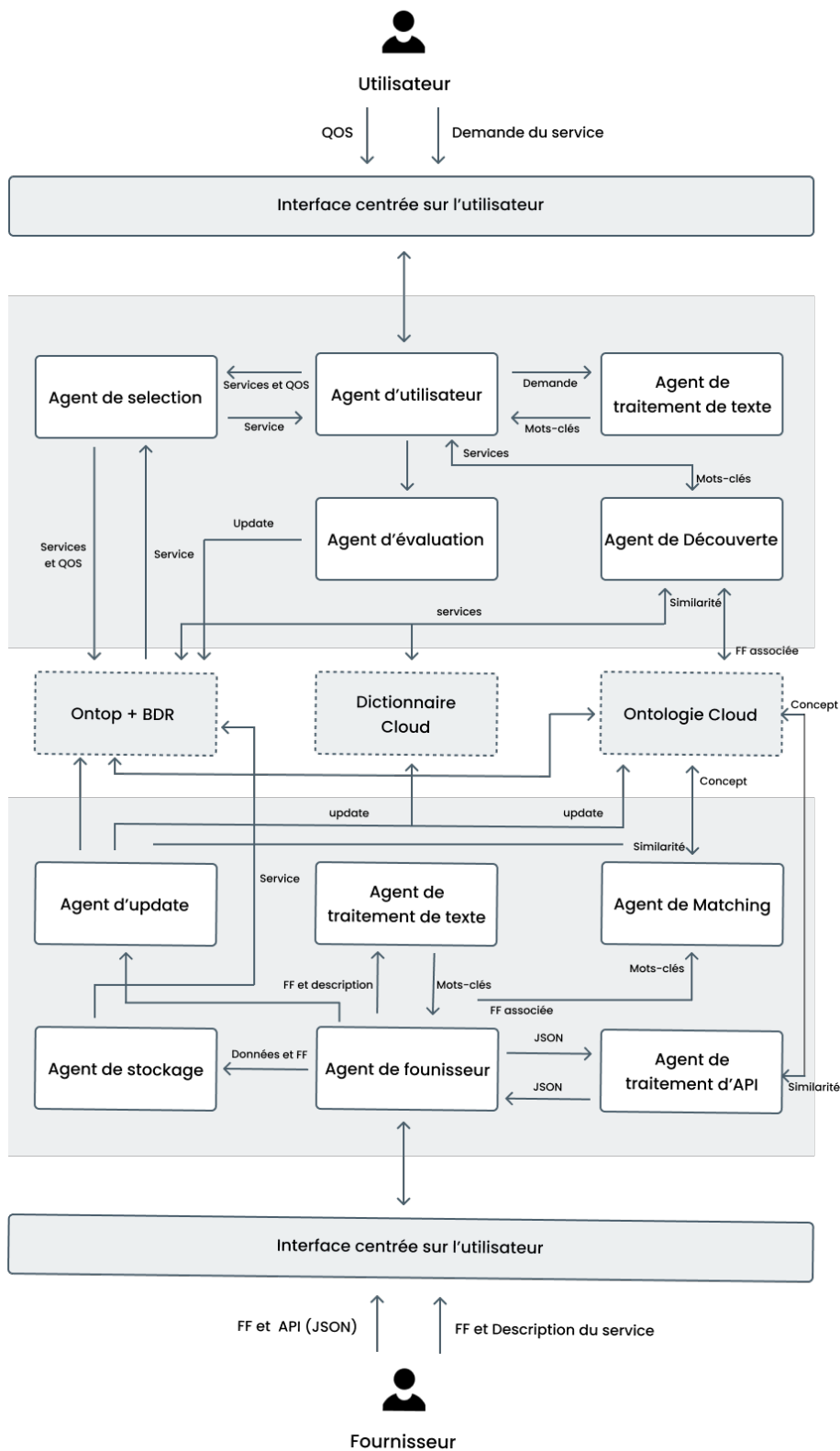


FIG. IV.2 : Architecture de la solution de découverte de service Cloud proposée

Notre architecture est un système constitué d'un ensemble d'agents pour différentes tâches, ainsi qu'une ontologie, un dictionnaire Cloud et aussi une BDR pour le stockage des services.

Comme indiqué dans la figure (IV.2), les agents utilisés dans le SMA sont classés en deux catégories :

1. Les agents agissant pour le compte du fournisseur des services :

- **Agent de fournisseur** : L'agent de fournisseur fournit une interface graphique au fournisseur pour ajouter un service, il gère aussi la communication entre tous les agents du fournisseur.
- **Agent de traitement de texte** : Le rôle de cet agent est d'extraire les expressions clés à partir d'une demande de FF et de description de service de fournisseur.
- **Agent de traitement d'API** : Cet agent reçoit L'API entré par le fournisseur en format JSON dont le but d'identifier les paramètres de description d'API selon la description d'ontologie.
- **Agent de Matching** : L'agent de Matching reçoit les mots-clés via l'agent fournisseur et passe une recherche sur la FF dans l'ontologie, dans le but de trouver la FF associé à la demande de fournisseur.
- **Agent de stockage** : L'agent de stockage reçoit la FF ainsi les données via l'agent fournisseur et enregistre le service dans la BD.
- **Agent d'update** : C'est l'agent responsable de maintenir les données de l'ontologie et de dictionnaire Cloud à jour.

2. Les agents agissant pour le compte d'utilisateur de Cloud :

- **Agent d'utilisateur** : L'agent utilisateur communique avec l'interface d'utilisateur pour récupérer la requête, il gère aussi la communication entre tous les agents d'utilisateur.
- **Agent de traitement de texte** : Cet agent reçoit la requête entré par l'utilisateur pour préparer une liste des mots clés de cette requête et envoyer la liste des mots clés à l'agent utilisateur.
- **Agent de découverte** : L'agent de découverte reçoit les mots-clés à travers l'agent utilisateur, il consiste à découvrir les services existents qui correspondent à la demande de l'utilisateur du Cloud, après il récupère tous les services correspondant et envoie le résultat à l'agent utilisateur.
- **Agent de sélection** : L'agent de sélection consiste à sélectionner parmi les services récupérés lors de la découverte, le service le plus approprié à la demande

de client en utilisant les critères de QoS.

- **Agent de d'évaluation** : Lorsqu'un client évalue son service l'agent d'évaluation est le responsable de la modification de l'évaluation de service dans la BD.

IV.4.1 Ontologie Cloud :

Nous nous sommes inspirées des solutions [2], [50] et [44] pour réaliser notre ontologie en utilisant l'outil Protégé. Cette ontologie fournit une classification sémantique des services Cloud en fonction de leur FF en plusieurs domaines, chaque domaine contient une hiérarchie des FF. De plus, les NFFs de ces services sont considérés dans cette ontologie pour rendre le processus de classement des services qui fournissent les mêmes FFs une opération efficace et pour unifier la description des services enregistrés par les fournisseurs.

Les concepts de notre ontologie sont illustrées dans la figure (IV.3) :

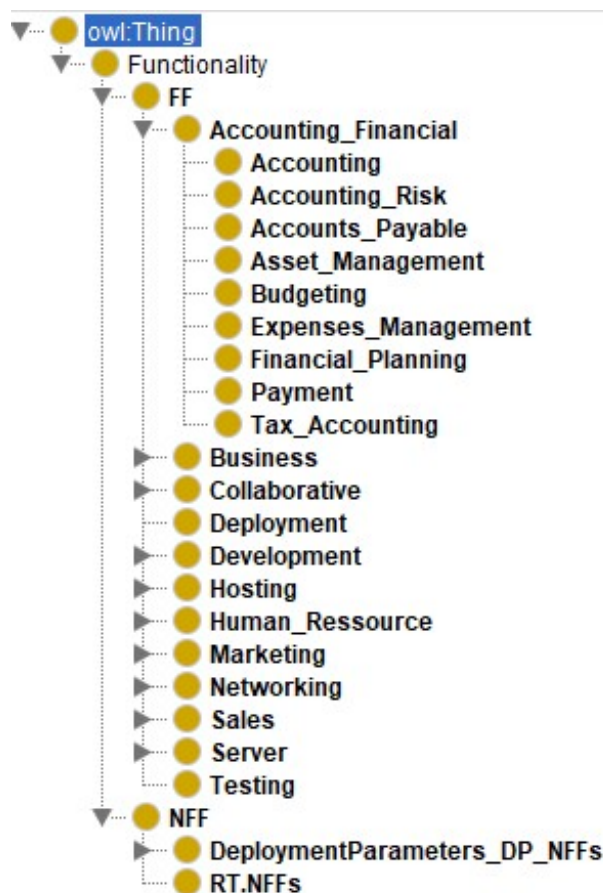


FIG. IV.3 : Les concepts des fonctionnalités des services selon notre ontologie proposée

- **Le concept des FFs** : Comprends les fonctionnalités publiées en tant que services individuellement ou dans des compositions pour répondre aux exigences fonctionnelles des utilisateurs du Cloud.

Les services qui fournissent plus que FF sont classés sous chaque FF qu'ils fournissent (illustré dans la figure (IV.3)).

- **Le concept des NFFs** :

On trouve :

- **Les fonctionnalités non fonctionnelles de déploiement (DeploymentParameters_DP_NFFs)** : Comme le montre la figure (IV.4), le concept DeploymentParameters_DP_NFFs implique 7 catégories principales.

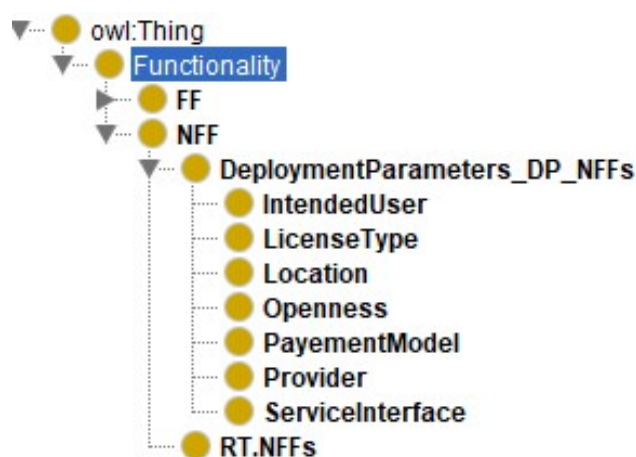


FIG. IV.4 : La catégorie des DPs selon l'ontologie proposée

Ce Concept inclut les NFFs généraux, pour chaque service Cloud. Comme indiqué dans le tableau suivant :

Data type properties	Domains	Ranges
ServiceTitle	DeploymentParamaters_DP_NFFs	String
ServiceURL	DeploymentParamaters_DP_NFFs	String
ShortDescription	DeploymentParamaters_DP_NFFs	String
SLA	DeploymentParamaters_DP_NFFs	String
SLATokens	DeploymentParamaters_DP_NFFs	String
Version	DeploymentParamaters_DP_NFFs	String

TAB. IV.1 : Les propriétés de DP selon notre ontologie proposée

- **Les fonctionnalités non fonctionnelles au moment de l'exécution (RT.NFFs) :**

Les clients ayant une faible expérience de la technologie Cloud peuvent avoir besoin de certains paramètres pour faciliter leur évaluation. Selon notre enquête, il existe un ensemble de paramètres communs pour faciliter l'évaluation de la qualité des services fournis (QoS). Comme illustré dans le tableau (IV.2).

Data type properties	Domains	Ranges
Price	RT.NFFs	Double
Rating	RT.NFFs	Double
Availability	RT.NFFs	Integer

TAB. IV.2 : Les propriétés de RT selon notre ontologie proposée

• **Création du tableau des relations :**

Dans le tableau ci-dessous (IV.3), nous décrivons les différentes relations binaires existantes (Object Properties) entre les différents concepts présentés :

Object properties	Domains	Ranges
ProviderID	DeploymentParamaters_DP_NFFs	Provider
LocationID	DeploymentParamaters_DP_NFFs	Location
LicenseTypeID	DeploymentParamaters_DP_NFFs	LicenseType
IntendedUserID	DeploymentParamaters_DP_NFFs	IntendedUser
OpennessID	DeploymentParamaters_DP_NFFs	Openness
PayementModelID	DeploymentParamaters_DP_NFFs	PayementModel
RT_NFFsID	RT	RT.NFFs
ServiceInterfaceID	DeploymentParamaters_DP_NFFs	ServiceInterface
DP_NFFsID	FF	DeploymentParamaters_DP_NFFs

TAB. IV.3 : Les relations entre les concepts de notre ontologie proposée

IV.4.2 Dictionnaire Cloud

Le dictionnaire Cloud englobe des termes anglais et leurs synonymes qui composent les descriptions textuelles des services Cloud les plus existants pour chaque FF à l'aide de BabelNet. Pour réaliser ce dictionnaire nous nous sommes inspirées de la solution [2].

Les termes et leurs synonymes seront stockés dans le dictionnaire au format JSON. pour chaque terme nous avons le type de terme, la liste des concepts, la liste des entités et le nom de domaine Cloud (FF).

Le rôle principal de ce dictionnaire est d'éviter l'ambiguïté de sens où les synonymes sont collectés pour chaque terme puis classer dans des domaines Cloud pour éviter les homonymes, et aussi pour ajouter un aspect sémantique de plus de l'ontologie.

IV.4.3 BDR

C'est une BD pour stocker les instances et les valeurs de données liées aux NFFs des services Cloud. Dans le but de gérer (stocker et interroger) une telle quantité de données avec une haute performance.

Dans notre solution les connaissances sont représentées par deux modèles : **Modèle BDR** et **modèle d'ontologie**. Les deux modèles ont des forces et des faiblesses. L'objectif principal des ontologies est de fournir un modèle de représentation sémantique des données avec un accent minimal sur la structure des instances de stockage. D'autre part, l'objectif principal des BDR est de fournir un référentiel bien structuré pour stocker efficacement les instances en mettant l'accent sur la signification sémantique des données. En d'autres termes, les ontologies se concentrent sur la signification des données, tandis que les BDR se concentrent sur le stockage des données. Bien que les ontologies puissent surmonter des problèmes difficiles, tels que l'interopérabilité, la recherche et la communication humaine en partageant les connaissances et la compréhension, elles ne parviennent pas à gérer une grande quantité de données.

IV.4.4 Accès aux données OBDA basé sur Ontop

C'est une couche intermédiaire entre l'ontologie Cloud et la BDR. L'outil Ontop est utilisé pour combiner les deux modèles. La fonctionnalité principale de OBDA est la réponse aux requêtes et transfert des requêtes SPARQL en SQL. Elle contient deux composants principaux :

- **Composant de règles de mappage** : La combinaison entre l'ontologie et la BDR est réalisée en générant un ensemble des règles de mappage pour lier les concepts ontologiques et leurs relations avec les tables BDR et leurs colonnes. Ces règles sont

responsables de la conversion des données stockées dans la source BDR en un ensemble de triplets RDF.

- **Composant de composition de requête Ontop SPARQL :** Pour interroger le modèle OBDA généré, la syntaxe Ontop SPARQL query est suivie. À partir du mapping créé on peut construire des requêtes SPARQL sur Ontop, ce dernier il convert les requêtes SPARQL en SQL, il applique les requêtes SQL sur BDR et il affiche les résultats.

IV.4.5 Fonctionnement du SMA

Cette section présente la méthodologie complète de notre travail, on distingue deux scénarios :

IV.4.5.1 Scénario de fournisseur :

Dans ce scénario nous définissons les différentes étapes depuis la réception de la requête du fournisseur jusqu'à l'enregistrement de son service.

- **Étape 1:** Les fournisseurs utilisent l'interface pour spécifier la FF de leurs services. Ils composent des requêtes en langage naturel. La requête de fournisseur est transmise à l'agent de fournisseur pour la transmission de message.
- **Étape 2:** A partir du moment où le fournisseur remplit sa requête de FF l'agent de fournisseur la transmet à l'agent de traitement de texte pour préparer un ensemble de mots clés en suivant les étapes indiquées dans la figure (IV.5), après l'agent de traitement de texte renvoie la liste des mots clés à l'agent de fournisseur. Les étapes de ce traitement de texte sont déjà expliquées dans la solution [2] (voir Annexe A).

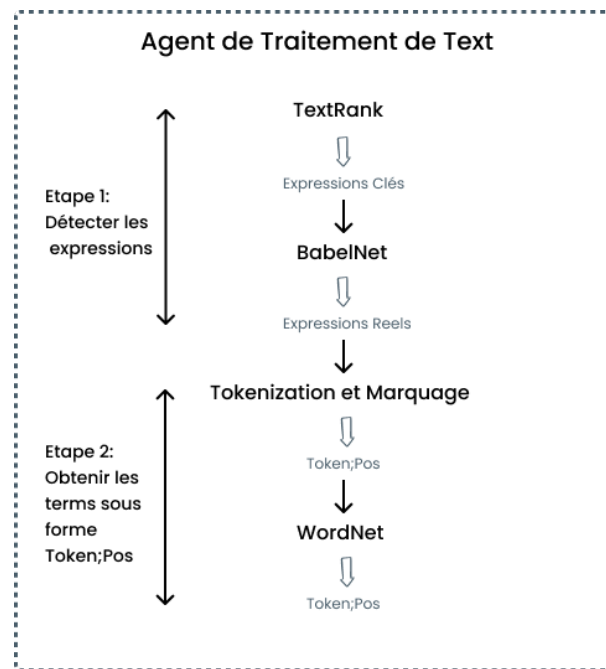


FIG. IV.5 : Le processus de Traitement de Texte

- **Étape 3:** L'agent de fournisseur envoie la liste des mots-clés à l'agent de Matching. L'agent de Matching consulte l'ontologie et calcule la similarité pour trouver d'abord le domaine précis aux mots-clés après il cherche dans l'arbre de domaine trouvé sur la FF adaptés à la liste des mots-clés qu'il a recevez (le calcul de similarité est présenté dans la section suivante). Lorsqu'il trouve la FF associée, l'agent de Matching envoie la FF associées à la l'agent de fournisseur pour la sauvegarder. Dans le cas où il ne trouve pas un domaine ou une FF pour la demande de FF de fournisseur il rajoute la FF demandée dans l'ontologie et envoyer à l'agent update pour créer un modèle ontop et une table dans BDR pour la FF.
- **Étape 4:** Dans l'étape suivante le fournisseur choisira de saisir son service via l'API ou via des champs à remplir.
- **Étape 5:** Dans le cas où il utilise une API, Le fournisseur utilise l'interface pour spécifier URI de son API en format JSON. Le lien de son API sera transmis à l'agent de fournisseur pour la transmission de message.
- **Étape 6:** L'agent de fournisseur transmet l'API à l'agent de traitement d'API pour calculer la similarité entre les paramètres de description d'API et la description d'ontologie (Par exemple calculer la similarité entre le paramètre 'nom du service' et le paramètre 'titre du service' dans l'ontologie). Après le calcul de similarité l'agent de

traitement d'API collecte la description de son service dans un fichier JSON et envoie le résultat à l'agent de fournisseur.

- **Étape 7:** Dans le cas où il utilise une description de service à travers des champs à remplir, le fournisseur utilise l'interface pour spécifier les paramètres de son service. Et envoie la description de son service à l'agent de fournisseur.
- **Étape 8:** L'agent de fournisseur envoie la description de service entrée à l'agent de traitement de texte pour collecter la liste des SLATokens qui contiennent les différents mots clés de description de service.
- **Étape 9:** L'agent de traitement de texte envoie les SLATokens à l'agent de fournisseur. Dans cette étape l'agent de fournisseur à son tour envoie les données de service ainsi que la FF à l'agent de stockage.
- **Étape 10:** L'agent de stockage stocke les services dans la BD dans la FF spécifiée.
- **Étape 11:** Après chaque stockage d'un nouveau service l'agent Update met les données à jour. Il récupère tous les FF qui ont des services dans notre BD, ainsi il récupère tous les SLATokens des services après il cherche les mots clés qui se répètent le plus dans notre SLATokens. Ces mots clés doivent représenter les FF dans notre ontologie. Donc on va mettre à jour notre BD ainsi que les mots clés qui représentent les FF dans notre ontologie on doit les enregistrer dans le dictionnaire Cloud, en suivant l'étape indiquée dans la figure (IV.6) ces étapes sont déjà expliquées dans la solution [2] (voir Annexe A).

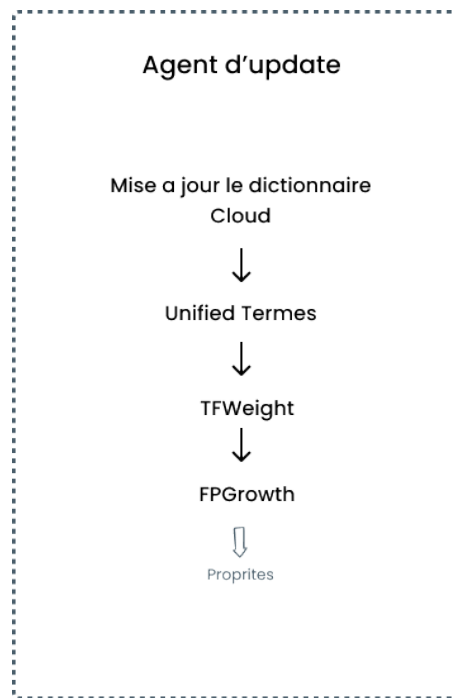


FIG. IV.6 : Le processus de mise à jour des mots clés associés

IV.4.5.2 Scénario de client :

Dans ce scénario nous définissons les différentes étapes depuis la réception de la requête jusqu'au retour du résultat au utilisateur.

- **Étape 1:** L'utilisateur utilise l'interface pour découvrir les services Cloud. Ils commencent leur processus de découverte en composant des requêtes de découverte de texte détaillées en utilisant des termes familiers en langage naturel anglais, ces termes en les fournissant à l'interface utilisateur en tant d'entrées. La demande de l'utilisateur est transmise à l'agent d'utilisateur pour la transmission de message.
- **Étape 2:** A partir du moment où l'utilisateur remplit sa requête de recherche l'agent d'utilisateur la transmet a l'agent de traitement de texte. Dans cette étape l'agent de traitement de texte envoie les entrées au processus Traitement de Texte en suivant les étapes indiquée dans la figure (IV.5), Après l'agent de traitement de texte renvoie la liste des mots clés a l'agent d'utilisateur. Les étapes de ce traitement de texte sont déjà expliquées dans le chapitre la solution [2] (voir Annexe A).
- **Étape 3:** L'agent d'utilisateur envoie la liste des mots clés à l'agent de découverte. L'agent de découverte consulte l'ontologie et calcule la similarité pour trouver les FF adaptés à la liste des mots clés qu'il a recevez (le calcul de similarité est présenté dans

la section suivante). Lorsqu'il trouve la FF associée l'agent de découverte consulte la BDR pour récupérer la liste du service pertinent à la FF. Une fois que la découverte réussie des services Cloud appropriés l'agent de découverte envoie les informations à la l'agent d'utilisateur. Enfin, les services Cloud et leurs NFFs sont récupérés sur l'interface utilisateur.

- **Étape 4:** Lorsque l'utilisateur reçoit la liste des services, il peut sélectionner le meilleur service parmi les services, il choisit les valeurs des critères avec le poids de chaque critère.
- **Étape 5:** L'agent d'utilisateur transmet les services et les exigences QoS saisis par l'utilisateur à l'agent de sélection pour l'étape de sélection de meilleur service. Dans cette étape l'agent de sélection consulte la BD pour sélectionner parmi les services envoyés par l'agent d'utilisateur le service le plus approprié aux critères QoS choisis par le client (disponibilité, réputation et le prix). Dans cette étape nous avons implémenté la sélection par MCDM.
- **Étape 6:** Le meilleur service choisit par l'agent de sélection sera transmis à l'agent d'utilisateur pour l'affichage du service dans l'interface d'utilisateur.
- **Étape 7:** Le client peut se connecter pour évaluer son service, l'agent d'utilisateur transmet la note d'évaluation à l'agent d'évaluation.
- **Étape 8:** L'agent d'évaluation travaille pour modifier la valeur de réputation du service dans la BD.

IV.4.6 Simulation sur le calcul de similarité

Dans cette section nous présentons les résultats obtenus des différentes mesures qu'on a testé et utilisé pour calculer la similarité entre 2 requêtes de description de service et 2 demandes de FF de fournisseur et les concepts d'ontologie, et entre une requête de recherche d'utilisateur avec les concepts de notre ontologie. Nous avons pris l'exemple des domaines montrés dans les figures (IV.7) (IV.8).

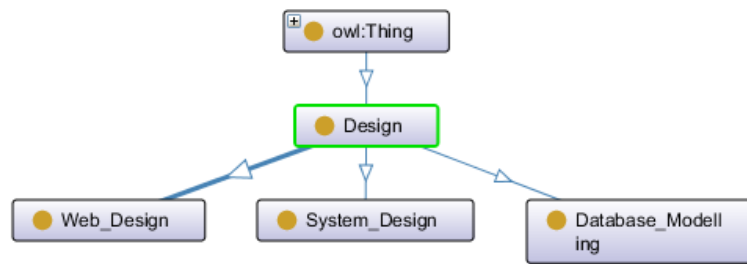


FIG. IV.7 : Domaine de Design



FIG. IV.8 : Domaine de Accounting and Financial

1. Requêtes de description des services de fournisseur :

- **Requête 1:**

La requête 1 représente la description de service **DeZign for Databases**², dans la fonctionnalité est **Database Modelling** dans l'ontologie (IV.7) :

Requête 1 = "Visualize database structures to understand your database, create databases or modify existing databases, you can also analyze and optimize databases."

Les expressions clés extraire a partir de cette description sont :

Expressions-clés = {Visualize database structures, databases}

Pour calcule la similarité entre une requête et un concept d'ontologie on calcule la similarité entre chaque expression clé de requête avec un concept après la similarité global c'est la moyenne de similarités des expressions de requête avec le concept.

Le tableau suivant montre la similarité entre la requête 1 et les concepts d'ontologie de la figure (IV.7) :

2. <https://www.datanamic.com/dezign/>

Concept (FF)	Dice	Jaccard	Levenshtein	Lexicale	Wu_Palmer	Sémantique
Database_Modelling	0.64	0.45	0.33	0.52	0.0	0.15
System_Design	0.11	0.23	0.18	0.03	0.33	0.32
Web_Design	0.09	0.26	0.11	0.03	0.67	0.5

TAB. IV.4 : Comparaison entre les différentes mesure de similarité et Requête 1

Etant donné les deux concepts « databases » et « Database Modelling », la similarité syntaxique est calculée selon le principe ci-dessus :

Les paires de caractères des deux concepts composés respectifs sont :

« **databases** » : da, at, ta, ab, ba, as, se, es = 8 paires

« **Database Modelling** » : da, at, ta, ab, ba, as, se, mo, od, de, el, ll, li, in, ng = 15 paires

Le résultat de l'intersection entre les paires de caractères est : da, at, ta, ab, ba, as, se = 7 paires

Si on prendre la distance de **Dice** comme exemple :

$$S_{Dice}(databases, DatabaseModelling) = \frac{2 * 7}{23} = 0.6$$

Grâce à cette valeur de 0,6 il est possible de déterminer que ces deux mots sont à peu près semblables.

On utilise le même principe avec les deux concepts « Visualize database structures » et « Database Modelling » :

$$S_{Dice}(Visualizedatabasestructures, DatabaseModelling) = \frac{2 * 8}{23} = 0.69$$

On calcule la Similarité Totale :

$$S_{Dice}(Requete1, DatabaseModelling) = 0.6 + 0.69/2 = 0.64$$

Pour le calcul de la mesure lexical entre les concepts des ontologies et les termes des requêtes, on a utilisé la formule mentionnées (1) qui calcule la similarité entre deux termes en se basant sur les synsets de chaque terme en utilisant WordNet.

Et pour le calcul de la mesure structurelle on a utilisé la mesure **Wu_Palmer** de Wordnet par le principe suivant :

Comme WordNet n'accepte que des termes simples on a divisé les termes complexes

des requêtes ont mots, la même chose pour les concepts des ontologies complexe, après on a calculé la similarité entre chaque mot, ensuite on a choisi le max de similarité des mots comme une similarité de terme de requête. Après on a calculé la moyenne des similarités pour trouver la similarité structurelle globale.

- **Requête 2:**

La requête 2 représente la description de service **Tax Accounting**³, dans la fonctionnalité est **Tax_Accounting** dans l'ontologie (IV.8) :

Requête 2 = "Our services help you get the right numbers by performing tax accounting calculations, investigating technical issues and reviewing transactions, accounting entries and adjustments that could have tax consequences."

Les expressions clés extraire a partir de cette description sont :

Expressions-clés = {tax accounting calculations, technical issues, transactions, adjustments, entries, numbers}

Le tableau suivant montre la similarité entre la requête 2 et les concepts d'ontologie de la figure (IV.8) :

Concept (FF)	Dice	Jaccard	Levenshtein	Lexicale	Wu_Palmer	Sémantique
Asset_Management	0.22	0.40	0.28	0.1	0.0	0.13
Budgeting	0.18	0.22	0.36	0.11	0.15	0.13
Tax_Accounting	0.27	0.41	0.29	0.2	0.0	0.14
Accounting_Risk	0.25	0.34	0.25	0.01	0.0	0.09
Accounts_Payable	0.18	0.39	0.20	0.09	0.15	0.14

TAB. IV.5 : Comparaison entre les différentes mesure de similarité et Requête 2

D'après les résultats mentionnés dans les deux tableaux nous pouvons remarquer que la mesure de **Dice** montre des résultats plus élevés par rapport à **Jaccard** et **Levenshtein**, notre choix s'est porté sur la mesure de Dice car elle nous aide à améliorer nos résultats de la similarité globale. Nous pouvons remarquer aussi que la mesure lexicale mentionnée dans (1) montres des résultats supérieurs pour les deux requêtes. La mesure **Wu_Palmer** donne des mauvais résultats pour les deux requêtes. Donc la similarité sémantique finale basée sur les trois mesures syntaxique, lexicale et structurelle donne de mauvais résultats à cause de mesures lexicale et structurelle.

3. https://www.ey.com/en_gr/tax/accounting-risk-advisory-services

Donc on ne peut pas utiliser cette mesure de similarité pour calculer la similarité entre la description de service de fournisseur et les concepts d'ontologie pour trouver la FF approprié a la description de service.

2. Requêtes de demande de FF de fournisseur :

- **Requête 1:**

La requête 1 représente un exemple de demande de FF de service dans la fonctionnalité est **Database Modelling** dans l'ontologie (IV.7) :

Requête 1 ="database modeling."

Les expressions clés extraire a partir de cette description sont :

Mots-clés = {database, modeling}

Pour calcule la similarité entre une requête et un concept d'ontologie on calcule la similarité entre chaque mot clé de requête avec un concept, ensuit la similarité global c'est la moyenne de similarités entre les mots clés et le concept dans le de cas de similarité syntaxique et lexicale, la similarité global c'est le max de similarités entre les mots clés et le concept dans le cas de similarité structurel. À la fin la similarité totale c'est la similarité sémantique qui combine les trois types de similarité comme le montre l'algorithme (2).

Le tableau suivant montre la similarité entre la requête 1 et les concepts d'ontologie de la figure (IV.7) :

Concept (FF)	Dice	Jaccard	Levenshtein	Lexicale	Wu_Palmer	Sémantique
Database_Modelling	0.54	0.42	0.38	0.04	1	0.67
System_Design	0.0	0.21	0.15	0.0	0.35	0.14
Web_Design	0.0	0.25	0.1	0.0	0.35	0.14

TAB. IV.6 : Comparaison entre les différentes mesure de similarité et Requête 1

- **Requête 2:**

La requête 2 représente un exemple de demande de FF de service dans la fonctionnalité est **Tax_Accounting** dans l'ontologie (IV.8) :

Requête 2 ="Managing tax accounting"

Les expressions clés extraire a partir de cette description sont :

Expressions-clés = {tax, accounting}

Le tableau suivant montre la similarité entre la requête 2 et les concepts d'ontologie de la figure (IV.8) :

Concept (FF)	Dice	Jaccard	Lenvenshtein	Lexicale	Wu_Palmer	Sémantique
Asset_Mangement	0.04	0.23	0.12	0.0	0.45	0.19
Budgeting	0.17	0.18	0.25	0.0	0.0	0.06
Tax_Accounting	0.47	0.45	0.42	0.27	1	0.68
Accounting_Risk	0.38	0.33	0.25	0.12	1	0.64
Accounts_Payable	0.22	0.25	0.25	0.09	0.37	0.24

TAB. IV.7 : Comparaison entre les différentes mesure de similarité et Requête 2

D'après les résultats mentionnés dans les deux tableaux nous pouvons remarquer que la mesure de **Dice** montre des résultats plus élevés par rapport à **Jaccard** et **Lenvenshtein**, notre choix s'est porté sur la mesure de Dice car elle nous aide à améliorer nos résultats de la similarité globale. Nous pouvons remarquer aussi que la mesure lexicale mentionnée dans (1) montres des résultats supérieurs pour les deux requêtes. La mesure **Wu_Palmer** aussi donne des bonne résultats. Donc la similarité sémantique finale basée sur les trois mesures syntaxique, lexicale et structurelle donne une bonne résultats. Donc notre solutions basé sur ce principe.

3. Requête de recherche d'utilisateur :

La requête suivante représente une demande de recherche sur un service, dans la fonctionnalité est **Tax_Accounting** dans l'ontologie (IV.8) :

Requête = "I want a service that helps me to get the numbers right by preparing tax accounting calculations, researching technical issues and reviewing transactions, accounting entries and adjustments that may have tax consequences."

Les expressions clés extraire a partir de cette description sont :

Expressions-clés = {numbers right, tax accounting calculations, technical issues, transactions, entries, adjustments}

On utilise la mesure de similarité H_{rel} de la solution [2]. Comme nous avons parlé avant, la

solution [2] suppose que chaque FF est défini par deux types de mots clés, mots clés uniques et mots clés fusionnés. Les termes de la requête des utilisateurs sont mis en correspondance avec les mots clés uniques associés au concept FF. Aussi, les mots clés fusionnés associés aux enfants de ce concept sont mis en correspondance pour décider s'il existe d'éventuels concepts apparentés ou non afin de décider d'éliminer ou non la recherche dans son sous-réseau pour diminuer le temps de recherche (Nous avons déjà expliqué cette étape en détaille dans Annexe A). Les FFs qui atteignent $H_{rel} \geq 60\%$ sont ajoutés à la liste "Fonctionnalités correspondantes" pour être récupérés en tant que FF les plus liés à la requête d'utilisateur, après on choisit le max de similarité entre les FFs récupère comme la FF la plus liée a la demande d'utilisateur comme le montre les deux algorithmes (3) et (4). On suppose qu'il existe déjà des services dans la FF est **Tax_Accounting** dans notre BD, la figure (IV.9) représente les mots clés unique et marged de concept **Tax_Accounting**.

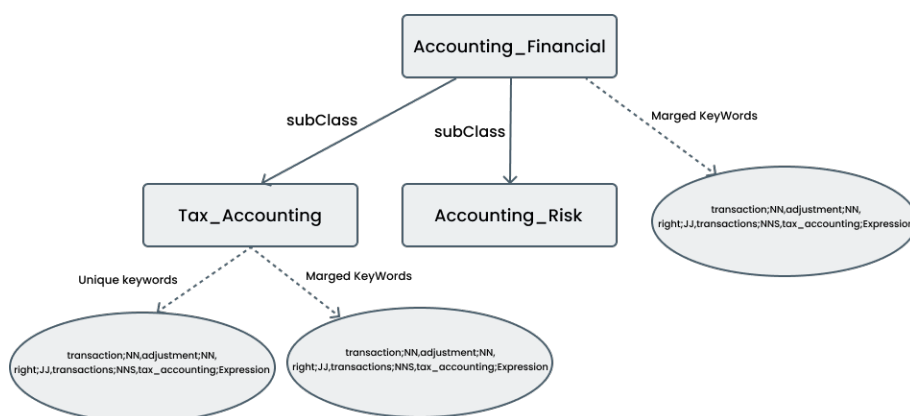


FIG. IV.9 : Exemple de keywords de concept **Tax_Accounting**

La valeur de H_{rel} entre la requête et les mots clés de concept **Tax_Accounting** = 0.79

Après plusieurs testes sur les similarités mentionnées, on a choisi pour notre solution le 2ème cas pour calculer la similarité entre FF entrer par le fournisseur et les concepts d'ontologie, et le 3ème cas pour calculer la similarité entre la requête d'utilisateur et les concepts d'ontologie.

IV.4.7 Sélection par MCDM

Après la découverte des service qui répondant à l'aspect fonctionnel de la demande du client, une sélection de service sera effectuée en utilisant la méthode MCDM. Notre sélection est basée sur les critères de qualité de service (QoS).

1. Les services avec les valeurs des QoS :

Le tableau suivant montre les valeurs de qualité de service de trois services dont la fonction est "Accounting" et qui peuvent satisfaire la requête de l'utilisateur.

<i>Service</i> \ <i>QoS</i>	Price	Availability	Rating
Service1	2.5	90	3
Service2	1.5	50	2.5
Service3	3.0	70	5

TAB. IV.8 : Les services candidats et les valeurs de leurs critères de QoS

2. La normalisation :

La normalisation des données est un processus de transformation pour obtenir des données d'entrée numériques et comparables en utilisant une échelle commune. Après avoir collecté les données d'entrée, nous devrions effectuer un pré-traitement pour le rendre utile pour la modélisation de la décision.

Pour les contraintes non fonctionnelles positives : Le rating et la disponibilité sont des contraintes quantitative, la formule est la suivante : $V_{ij} = (R_{ij}) - (\text{Min } R_j) / (\text{Max } R_j) - (\text{Min } R_j)$.

Pour les contraintes non fonctionnelles négatives : Le prix est un contrainte quantitative, la formule est la suivante : $V_{ij} = (\text{Max } R_j) - (R_{ij}) / (\text{Max } R_j) - (\text{Min } R_j)$.

	Price normalisé	Availability normalisé	Rating normalisé
Service1	0.33	1	0.20
Service2	1	0	0
Service3	0	0.5	1

TAB. IV.9 : La matrice de décision

3. Supposons que le client à choisi suivants $R = \{2, 66, 3.5\}$ D'où 2 est la valeur attribuée au prix et 66 pour la disponibilité et la valeur 3.5 pour le rating. Ces valeurs dois être normalisées. Donc le vecteur normalisé est : $R = \{0.67, 0.40, 0.40\}$.

4. Supposons que le client à choisi les poids suivants $W = \{1, 3, 2\}$ D'où 1 est le poids attribué au prix, 3 pour l'availability et 2 pour le rating. Ces valeurs dois être norma-

lisées, le poids est limité entre 1 et 3. Donc le vecteur de poids normalisé est : $W = \{1, 1, 0.5\}$.

5. Nous soustrayons le vecteur R de chaque ligne de la matrice de décision pour obtenir la matrice Weighted Difference (WD).

	Price normalisé - R_1	Availability normalisé - R_2	Rating normalisé - R_3
Service1	-0.34	0.60	-0.20
Service2	0.33	-0.40	-0.40
Service3	-0.67	0.10	-0.60

TAB. IV.10 : La matrice de décision après la soustraction

6. Ensuite on fait la multiplication avec le vecteur W qui représente le poids attribué pour chaque contrainte non fonctionnelle $W = \{1, 1, 0.5\}$.

	WD * W_1	WD * W_2	WD * W_3
Service1	-0.34	0.60	-0.10
Service2	0.33	-0.40	-0.20
Service3	-0.67	0.10	-0.30

TAB. IV.11 : La matrice de décision après la multiplication (1)

7. ensuite, on multiplie la matrice par le scalaire (-1), et on met chaque valeur comme la puissance de la fonction exponentielle. Appelée Exponential Weighted Difference (EWD),

	WD * W_1	WD * W_2	WD * W_3
Service1	1.40	0.55	1.11
Service2	0.72	1.49	1.22
Service3	1.95	0.90	1.35

TAB. IV.12 : La matrice de décision après la multiplication (2)

8. Enfin, on fait la somme de chaque ligne de la matrice .

Service	Somme de la ligne
Service1	3.06
Service2	3.43
Service3	4.20

TAB. IV.13 : La matrice EWD

Donc **Service 1** est le meilleur service avec le score le moins élevé selon MCDM.

IV.5 Conclusion

Dans ce chapitre, nous avons présenté la conception de notre solution Cloud Broker en tant que courtier entre les fournisseurs de type SaaS et les consommateurs.

Nous avons commencé par introduire la démarche de travail suivie, puis nous avons fait une analyse des besoins pour identifier les acteurs du système et leurs actions respectives. Ensuite nous nous sommes lancées dans la présentation de notre solution en commençant par son architecture suivie de ses modules. Nous avons expliqué en détail chacun des modules de description, de découverte et de sélection à l'aide des exemples illustratifs et des algorithmes pour permettre et faciliter la mise en œuvre de cette solution, dans le but de conclure les résultats qui confirment la validité de notre solution.

L'implémentation de la solution proposée sera décrite dans le chapitre suivant.

Chapitre V

Implémentation de la solution

V.1 Introduction

Au cours du processus de développement, l'implémentation d'un logiciel vient après un enchaînement de plusieurs étapes et son but principal est de réaliser un produit capable de résoudre les problèmes posés en utilisant des outils et des algorithmes.

Dans le chapitre précédent, nous avons présenté la conception de notre solution. Dans ce chapitre, nous détaillons l'aspect implémentation et réalisation. Ce chapitre va comporter une partie pour spécifier les ressources matérielles et logicielles utilisées ainsi que des imprimés écrans qui montrent chaque interface de notre système suivant leurs cas d'utilisation.

V.2 Ressources matérielles

Notre application va être réalisée sur une machine qui comporte les caractéristiques suivantes :

- **Marque** : HP
- **Processeur** : Intel Core i5
- **Disque dur** : Disque dur 500 Go
- **Ram** : 4.00 Go
- **Système d'exploitation** : Windows 10 Professionnel

V.3 Outils et environnement de développement

Dans cette section, nous allons mentionner les outils, les langages de programmation et les plateformes utilisées pour implémenter ce système.

Eclipse

Est un projet de la fondation Eclipse¹, et un IDE écrit en Java publié par IBM² qui cherche à créer un environnement de production de logiciels englobant toutes les activités de programmation, modélisation, conception, test et gestion de configuration, aussi vise à supporter tout langage de programmation [51].

JDK

JDK³ ou Java Development Kit est un environnement de développement développé par Oracle, qui permet aux utilisateurs de créer des applications, des applets et des composants en Java. Il contient un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que des outils pour compiler du code Java.

JavaEE

JavaEE⁴ est une plateforme qui permet la création d'applications web. Elle offre également des fonctionnalités qui sont particulièrement intéressantes dans le développement d'applications au sein des entreprises. Java EE regroupe un ensemble d'API, chacune conçue pour effectuer un traitement spécifique. Dans le cadre de notre application, nous avons choisi d'utiliser les API suivantes :

- **Servlet** : Une servlet est une classe Java, qui a la particularité de permettre le traitement de requêtes et la personnalisation de réponses.
- **JSP (Java Server Pages)** : Les pages JSP sont l'une des technologies les plus puissantes de la plate-forme Java EE et sont faciles à utiliser et à mettre en œuvre. Ils se présentent sous la forme de simples fichiers au format texte avec des balises respectant la syntaxe. Le langage JSP combine les technologies HTML, XML, servlet et JavaBeans dans une seule solution.
- **JDBC (Java DataBase Connectivity)** : C'est une API qui fait partie intégrante de la plate-forme Java, qui se compose de classes qui permettent d'accéder à partir

1. www.eclipse.org

2. www.ibm.com

3. <https://www.oracle.com/java/technologies/downloads/>

4. <https://docs.oracle.com/javaee/7/tutorial/overview.htm#BNAAW>

d'applications Java à des données stockées dans des bases de données.

- **API Jena** : Jena⁵ est une API Java open source adoptée par la fondation apache. Elle entre en jeu dans la création d'applications du Web Sémantique. Cette API fournit des méthodes pour créer, interroger et manipuler des données en RDF, RDFS et OWL en utilisant le langage SPARQL et selon les recommandations du W3C.
- **OWL API** : L'API OWL⁶ est une API Java et une implémentation de référence pour créer, manipuler et sérialiser des ontologies OWL.
- **JAWS API** : Est une API qui permet aux utilisateurs d'ajouter à leur applications la possibilité de rechercher et de récupérer de données depuis la base de données Wordnet.

Tomcat

Tomcat⁷ est un logiciel open source qui représente les implémentations des technologies Java Servlet, JSP (Java Server Pages), Java Expression Language et Java WebSocket, publié sous la licence Apache.

Ce serveur est particulièrement utilisé pour le déploiement d'applications développées selon le modèle MVC (modele, view, Controller) et se basant sur les technologies de Servlet et de JSP (java server pages). Il permet de produire une représentation HTML à une requête après avoir effectué un certain nombre d'actions tel que la connexion a une base de données.

Protégé

Protégé⁸ est un editeur d'ontologie qui permet de construire des bases de connaissances. Il dispose d'une interface utilisateur personnalisable et d'outils de visualisation qui permettent une interaction avec l'ontologie et ses relations.

5. https://jena.apache.org/about_jena/about.html

6. <http://owlapi.sourceforge.net/>

7. <https://tomcat.apache.org/>

8. <https://protege.stanford.edu/products.php>

Xampp

XAMPP⁹ est une distribution Apache entièrement gratuite et facile à installer qui inclut MySQL, PHP et Perl. Il est donc à la portée du grand nombre de personne car il ne nécessite aucune connaissance particulières.

SQL

SQL¹⁰ (Structured Query Language) est un langage de requête à usage général et standardisé pour accéder et manipuler des bases de données au sein des SGBD et plus particulièrement des SGBDR. Les instructions SQL sont des instructions destinées à une base de données qui permettent aux utilisateurs d'effectuer diverses tâches. Les instructions SQL peuvent être utilisées directement par le système de gestion de base de données, mais elles peuvent également être intégrés dans du code écrit dans des langages tels que Java.

MySQL

Le système de gestion de base de données relationnelle (SGBDR) open source oracle MySQL¹¹ s'appuie sur le langage de requête structuré SQL. Il permet aux utilisateurs d'accéder et de traiter facilement les données stockées dans les différentes tables de la base de données. Il se caractérise par sa rapidité, son évolutivité, sa sécurité et sa simplicité d'utilisation.

HTML

L'HTML¹² est un langage utilisé pour créer des pages web. L'acronyme signifie HyperText Markup Langage. Cette signification porte bien son nom puisqu'effectivement ce langage permet de réaliser de l'hypertexte à base d'une structure de balisage.

9. <https://www.apachefriends.org/fr/index.html>

10. <https://docs.oracle.com/en/database/oracle/sql-developer/index.html>

11. <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>

12. <https://www.w3schools.com/html/>

CSS

Le terme CSS¹³ est l'acronyme anglais de Cascading Style Sheets. Le CSS est un langage informatique utilisé sur l'internet pour mettre en forme les fichiers HTML ou xml.

JavaScript

Le JavaScript¹⁴ est un langage informatique utilisé sur les postes client, en d'autres mots c'est votre ordinateur qui va recevoir le code et qui devra l'exécuter. C'est en opposition à d'autres langages qui sont activé côté serveur.

WordNet

WordNet¹⁵ est une base de données lexicales de relations sémantiques entre mots dans plus de 200 langues. Elle contient un ensemble d'entités, chaque entité est organisée en ensemble de synonymes (appelés synsets), reliées par différentes relations sémantiques. Chaque synset représente un concept de la langue anglaise. Il contient tous les mots synonymes pouvant exprimer le sens auquel il fait référence. Son but est de répertorier, classifier et mettre en relation de diverses manières le contenu sémantique et lexical de la langue anglaise. WordNet peut être utilisé dans notre mécanisme proposé pour récupérer les bases des mots.

BabelNet

BabelNet¹⁶ est un réseau sémantique qui connecte des concepts et des entités nommées dans une vaste base de connaissances de relations sémantiques. Le réseau comprend environ 20 millions d'entrées. Chaque entrée représente une signification donnée et son synonyme associé. BabelNet suit le modèle WordNet basé sur la notion de synset, chaque synset BabelNet représente une signification donnée et contient tous les synonymes qui expriment cette signification. Elle couvre 500 langues et est obtenue à partir de l'intégration automatique de : **WordNet, Open Multilingual WordNet, Wikidata, Wikipedia**, etc. Par conséquent,

13. <https://www.w3schools.com/css/>

14. <https://www.w3schools.com/js/>

15. <https://wordnet.princeton.edu>

16. <https://babelnet.org/about>

BabelNet peut être utilisé dans notre mécanisme proposé pour atteindre la convergence souhaitée entre la terminologie anglaise NL et les concepts d'ontologie Cloud.

Systeme Multi-Agent (SMA)

Un système multi-agents [3] est un système qui consiste en un groupe d'agents pour atteindre un objectif global ou différents objectifs. Basé sur des techniques de coordination et de négociation. Chaque agent est un système capable d'action autonome et réfléchi dans un environnement réel. Cela permet un comportement cohérent, optimisant la productivité des solutions émergentes. Les caractéristiques d'un SMA sont :

- **Fiabilité** : L'attribution de tâches à des agents spécifiques aide à résoudre les problèmes rapidement et efficacement.
- **Extensibilité** : L'indépendance des agents les uns par rapport aux autres facilite la modification de leurs comportements.
- **Autonomie** : Un agent peut agir sans revenir à son possesseur afin de prendre des décisions.
- **Réactivité** : Les agents peuvent percevoir puis réagir à divers événements dans leur environnement.
- **Raisonnement et rationalité** : L'agent est capable de raisonnement rationnel pour choisir la meilleure action pour optimiser sa productivité.

JADE¹⁷

C'est un framework qui permet le développement des systèmes multi-agents et d'applications conformes aux normes FIPA. La plateforme JADE fournit une interopérabilité sans limite pour les applications qu'elle prend en charge. JADE est composée de trois principaux modules qui dépendent directement des normes FIPA :

- **DF(Directory Facilitator)** : Fournit un service de pages jaunes à la plateforme.
- **ACC(Agent Communication Channel)** : Gère la communication entre les agents.
- **AMS(Agent Management System)** : Supervise l'enregistrement des agents, leur authentification, leur accès et l'utilisation de système.

17. <https://jade.tilab.com/>

V.4 Présentation de l'application

La réalisation de notre solution a pris la forme d'une application Web pour laquelle nous avons choisi le nom "CloudBroker". Nous avons aussi conçu un logo simple est significatif pour représenter notre application. Notre logo apparaît dans la figure suivante :



FIG. V.1 : Logo de CloubBroker

Le but principal de notre travail est de fournir une interface centrée sur l'utilisateur. Cette interface permet aux utilisateurs, qui manquent d'expérience et de connaissances dans le Cloud, de composer des requêtes de découverte complexes en utilisant la langue anglaise naturelle pour découvrir ou publier des services adaptés aux fonctionnalités informatiques prévues, il permet aussi aux utilisateurs d'obtenir directement des informations sur les services Cloud appropriés sans aucune intervention humaine.

Au début, lorsqu'un utilisateur tente d'accéder à notre application, via son navigateur Web, il atterrit dans la page de présentation du site. Comme représenté ci-dessous :

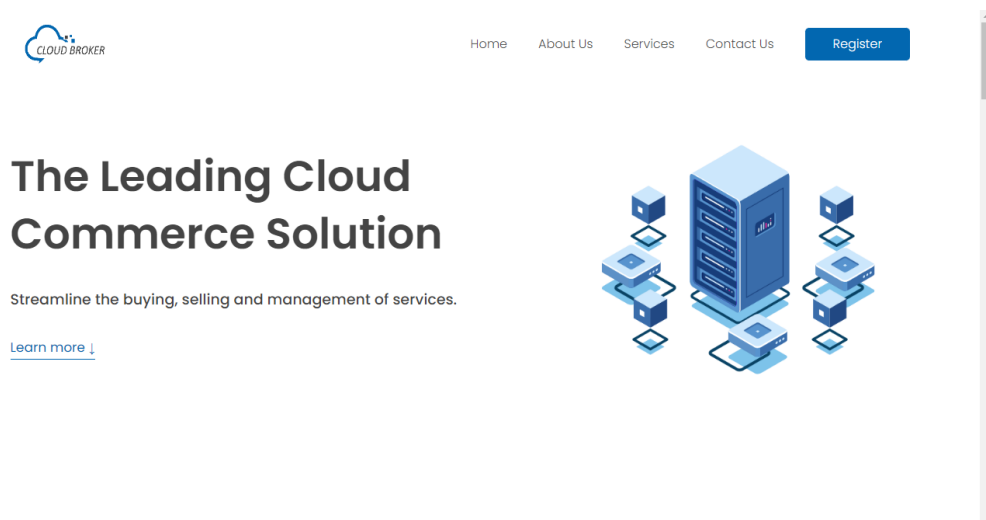


FIG. V.2 : Page de présentation du site

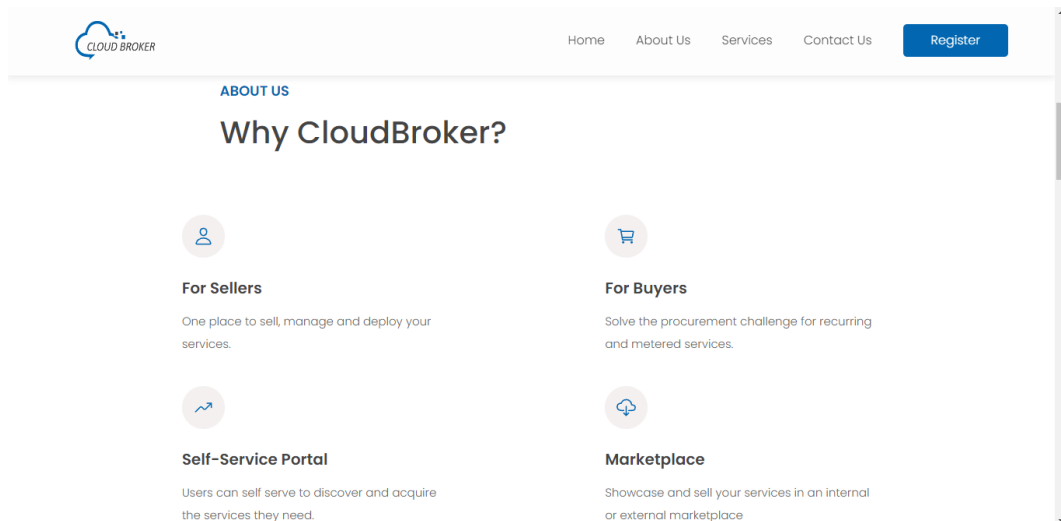


FIG. V.3 : Page de présentation du site (suite)

L'utilisateur peut se connecter s'il a déjà un compte. S'il n'est pas déjà inscrit, il peut créer un compte en cliquant sur lien "Create a new account?". Ce lien va le redigé vers l'interface d'inscription ou il doit introduire ses informations comme le montre la figure (V.6). L'utilisateur doit également préciser s'il souhaite rejoindre en tant que fournisseur ou en tant que client. Comme représenté ci-dessous :



FIG. V.4 : Type d'utilisateur

L'interface de connexion est illustré dans la figure (V.5).



FIG. V.5 : Interface de connexion

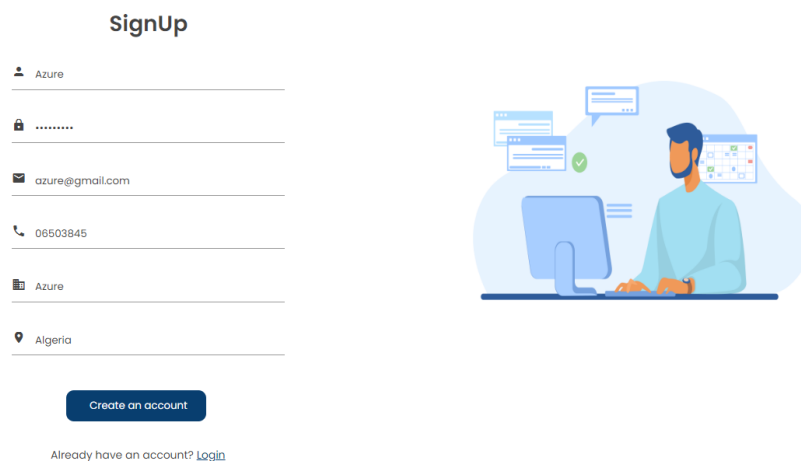


FIG. V.6 : Interface d'inscription

V.4.1 Espace Fournisseur

Une fois que le fournisseur se connecte, il est dirigé directement vers sa page d'accueil ou il peut choisir une tâche à effectuer, il peut consulter ses services, ajouter un service par API ou sans API il peut même voir les statistiques, ect. Comme représenté ci-dessous :

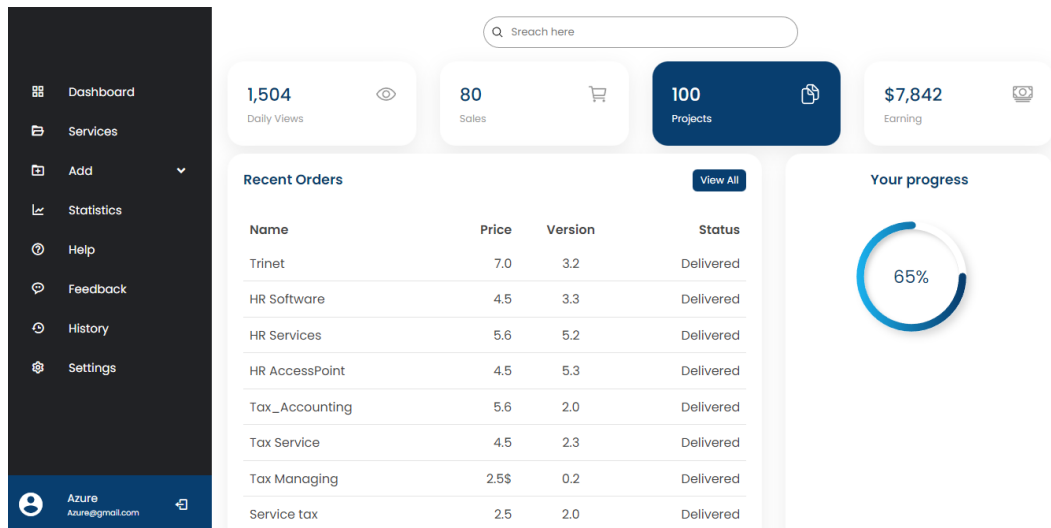


FIG. V.7 : Page d'accueil de fournisseur

En premier lieu, pour ajouter un service le fournisseur doit spécifier les FF en langage naturel comme le montre la figure (V.8).

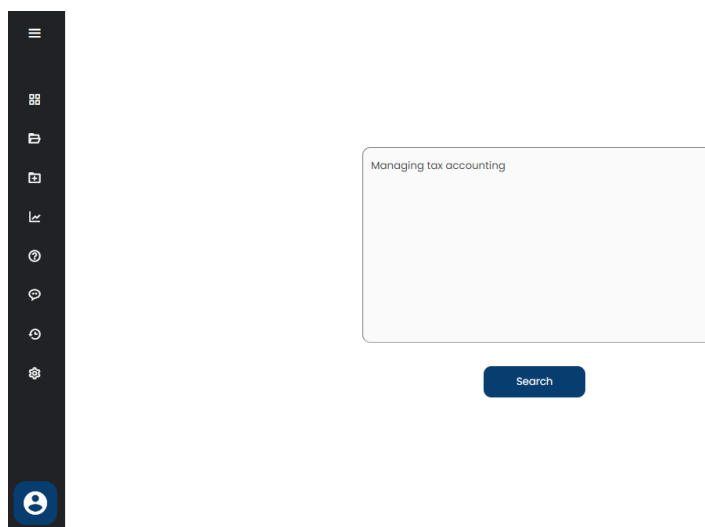


FIG. V.8 : Ajouter FF

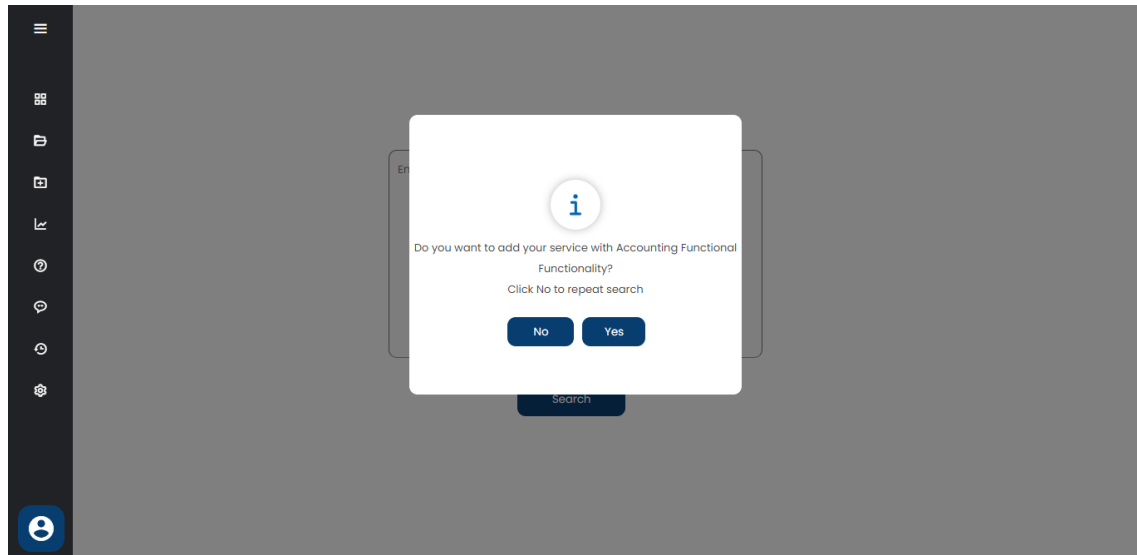


FIG. V.9 : Confirmation de FF

Ensuite, le système va chercher la FF similaire à la FF spécifiée par le fournisseur à partir de l'ontologie Cloud qui contient un ensemble des domaines et pour chaque domaine une hiérarchie des FF reliée par ce dernier comme le montre la figure (V.10).

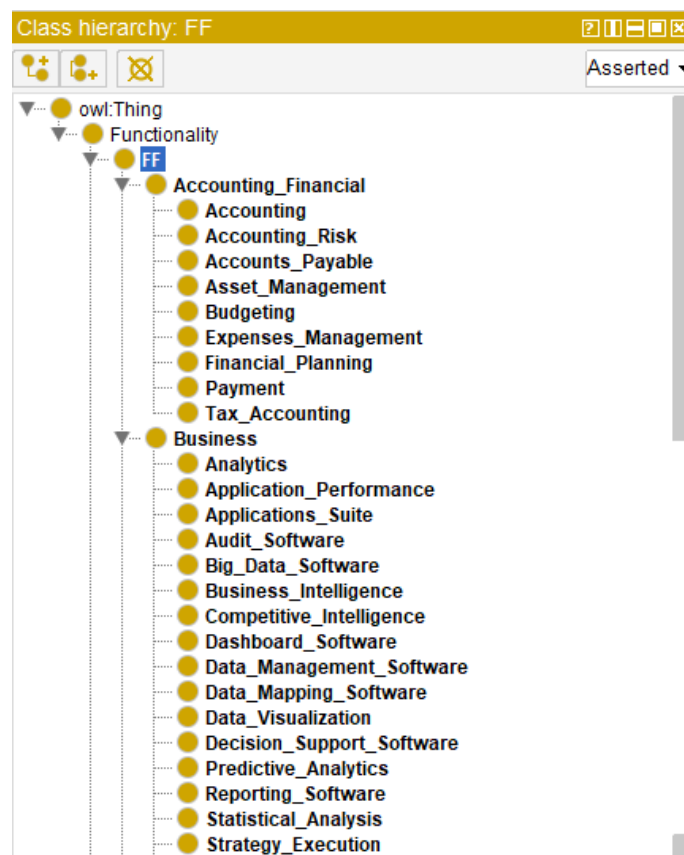


FIG. V.10 : Exemple d'ontologie Cloud

Après la saisie de la FF, le fournisseur a deux façons pour publier son Service, qui sont :

1. **Avec l'utilisation d'API :** Nous avons utilisé des API fake via **JSONServer**. Dans ce cas le fournisseur a la possibilité de saisir seulement le lien de son service à travers l'API (comme illustré dans la figure (V.11)) sans avoir besoin de remplir le formulaire de publication, ce lien contient la description de son service en format JSON (comme indiqué dans la figure (V.13)).

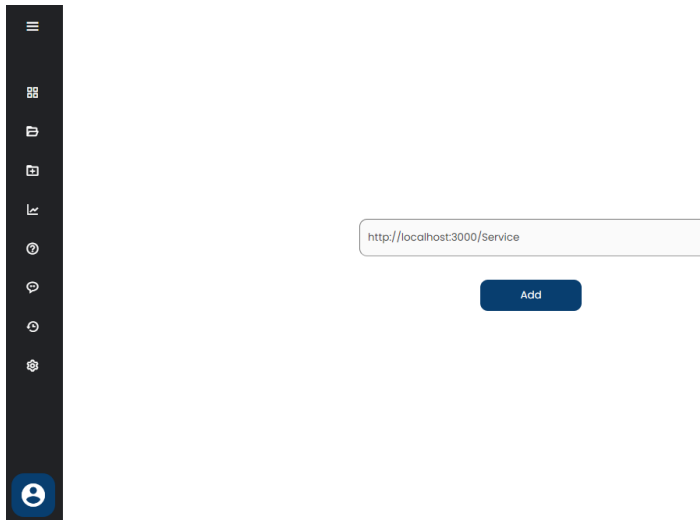


FIG. V.11 : Ajouter un service en utilisant l'API

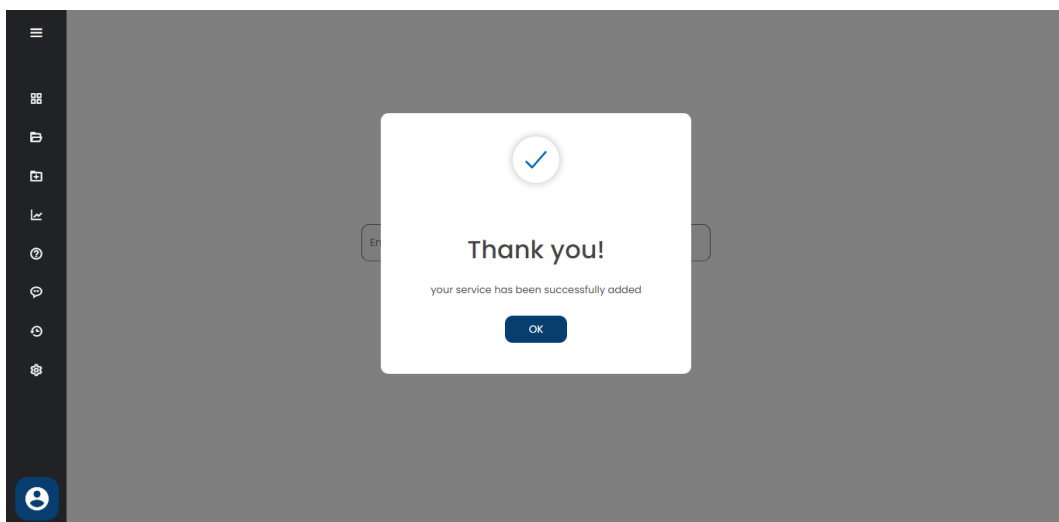
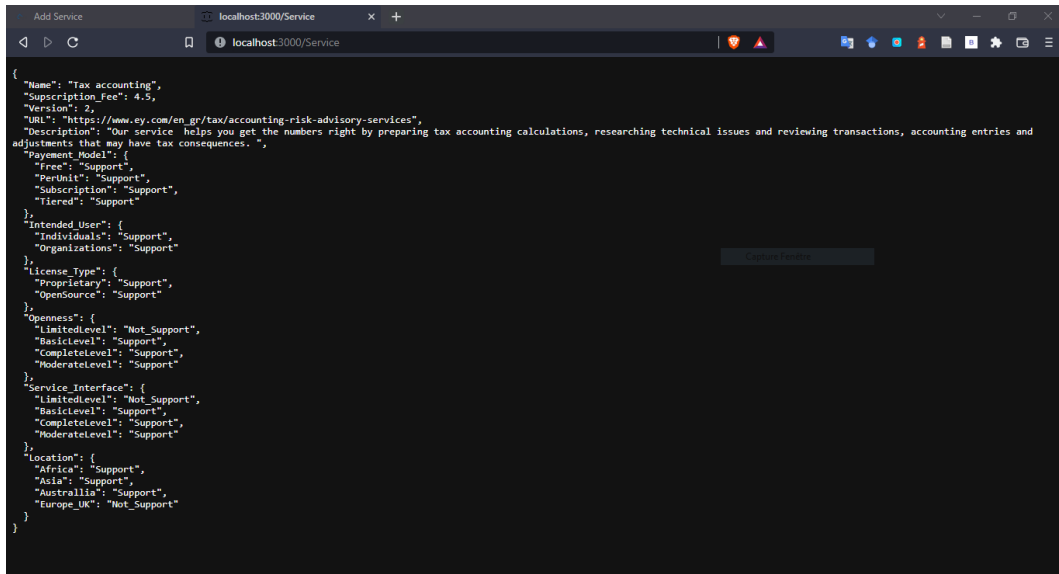
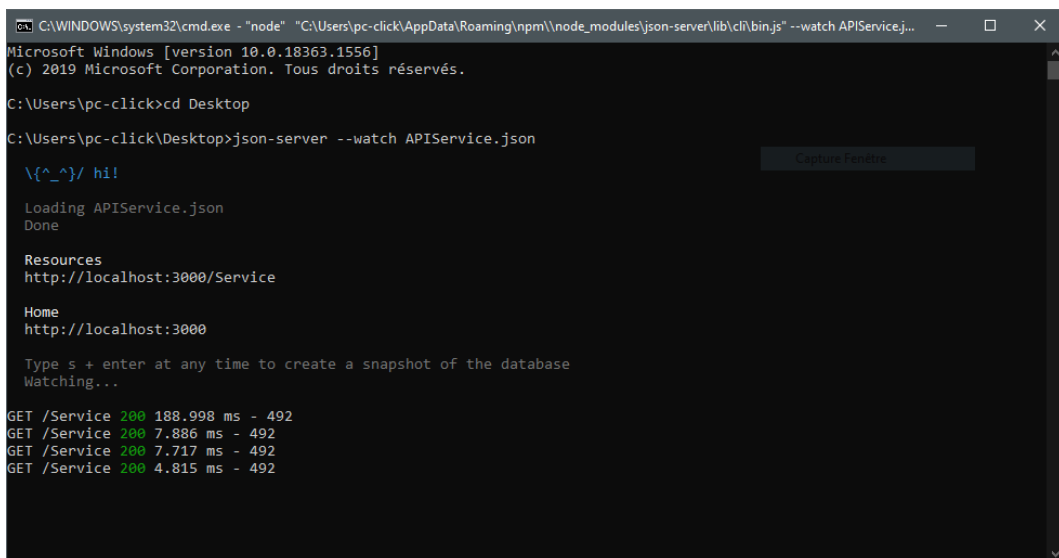


FIG. V.12 : Confirmer l'ajout du service



```
{
  "Name": "Tax accounting",
  "Subscription_Fee": 4.5,
  "Version": 2,
  "URL": "https://www.ey.com/en_gr/tax/accounting-risk-advisory-services",
  "Description": "Our service helps you get the numbers right by preparing tax accounting calculations, researching technical issues and reviewing transactions, accounting entries and adjustments that may have tax consequences.",
  "Payment_Model": {
    "Free": "Support",
    "Permit": "Support",
    "Subscription": "Support",
    "Tiered": "Support"
  },
  "Intended_User": {
    "Individuals": "Support",
    "Organizations": "Support"
  },
  "License_Type": {
    "Proprietary": "Support",
    "OpenSource": "Support"
  },
  "Openness": {
    "LimitedLevel": "Not_Support",
    "BasicLevel": "Support",
    "CompleteLevel": "Support",
    "ModerateLevel": "Support"
  },
  "Service_Interface": {
    "LimitedLevel": "Not_Support",
    "BasicLevel": "Support",
    "CompleteLevel": "Support",
    "ModerateLevel": "Support"
  },
  "Location": {
    "Africa": "Support",
    "Asia": "Support",
    "Australia": "Support",
    "Europe_UK": "Not_Support"
  }
}
```

FIG. V.13 : Exemple d'API



```
C:\WINDOWS\system32\cmd.exe - "node" "C:\Users\pc-click\AppData\Roaming\npm\node_modules\json-server\lib\cli\bin.js" --watch APIService.j...
Microsoft Windows [version 10.0.18363.1556]
(c) 2019 Microsoft Corporation. Tous droits réservés.

C:\Users\pc-click>cd Desktop
C:\Users\pc-click\Desktop>json-server --watch APIService.json

\{\^_}/ hi!

Loading APIService.json
Done

Resources
http://localhost:3000/Service

Home
http://localhost:3000

Type s + enter at any time to create a snapshot of the database
Watching...

GET /Service 200 188.998 ms - 492
GET /Service 200 7.886 ms - 492
GET /Service 200 7.717 ms - 492
GET /Service 200 4.815 ms - 492
```

FIG. V.14 : Configuration d'API en JSONServer

Après le système va calculer la similarité entre les paramètres de description d'API et la description d'ontologie (Par exemple calculer la similarité entre le paramètre 'nom du service' et le paramètre 'titre du service' dans l'ontologie).

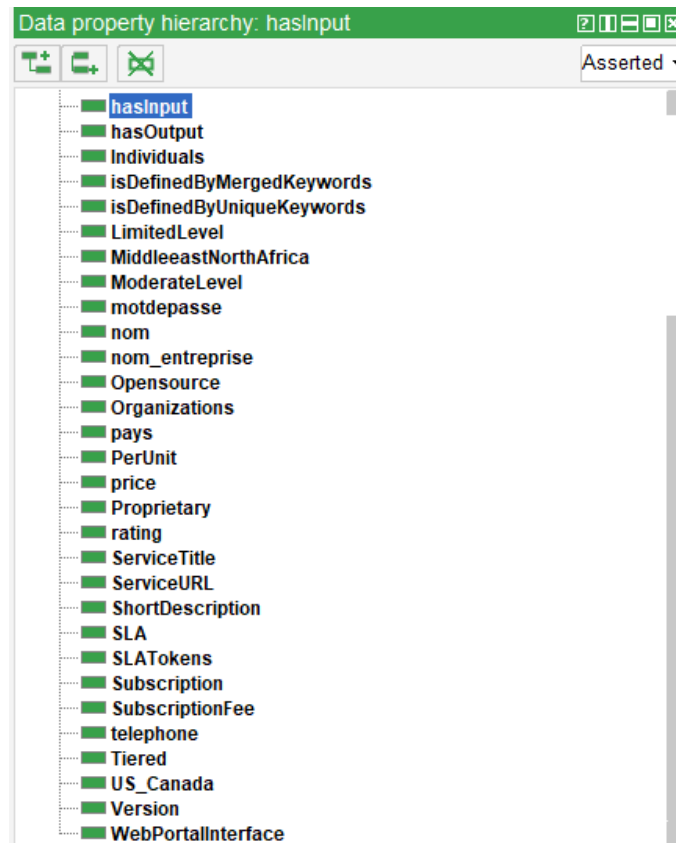


FIG. V.15 : Data properties

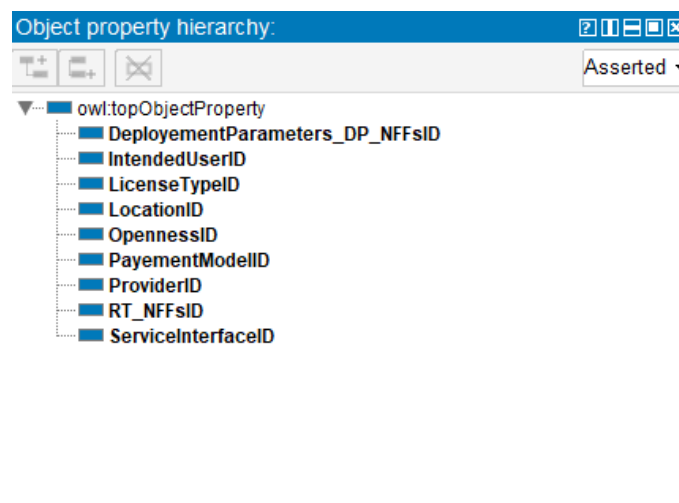


FIG. V.16 : Object properties

2. **Sans l'utilisation d'API** : Les figures (V.17), (V.18), (V.19) et (V.20) montrent l'interface de publication d'un service où le fournisseur doit saisir les informations relatives au service ainsi que les valeurs des critères de qualité de service.

The screenshot shows the 'General Parameters' section of a service publication interface. It includes a vertical sidebar on the left with various icons and a user profile icon at the bottom. The main content area contains the following fields:

- Service Title :** Tax Accounting
- Version :** 2.0
- Service URL :** https://www.ey.com/en_gr/ta
- Short Description :** Our service helps you to get the numbers right by preparing tax accounting calculations, researching technical issues and reviewing transactions
- Import SLA :** Choisir un fichier SLA.TXT
- Payment Model :**
 - Free :** Support
 - Tiered :** Support

FIG. V.17 : Interface de publication d'un service

The screenshot shows the 'Payment Model' and 'Openness' sections of a service publication interface. It includes a vertical sidebar on the left with various icons and a user profile icon at the bottom. The main content area contains the following fields:

- Payment Model :**
 - Free :** Support
 - Tiered :** Support
 - PerUnit :** Support
 - Subscription :** Support
- Openness :**
 - LimitedLevel :** Support
 - BasicLevel :** Not Support
 - CompleteLevel :** Not Support
 - ModerateLevel :** Support

FIG. V.18 : Interface de publication d'un service (Suite)

License Type :

Proprietary : Open Source :

Intended User :

Individuals : Organizations :

FIG. V.19 : Interface de publication d'un service (Suite)

Quality Of Service Paramaters :

Price : Availability :

FIG. V.20 : Interface des critères de QoS

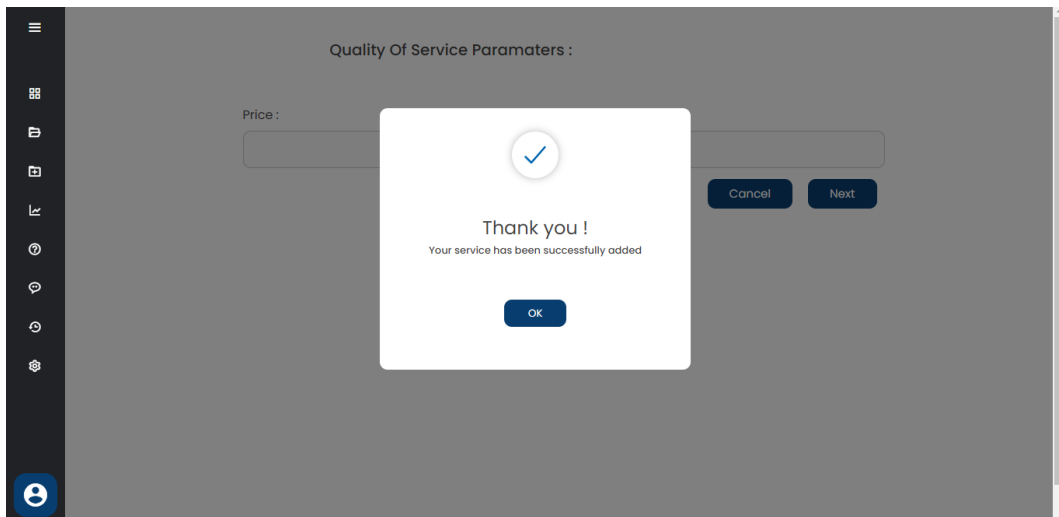


FIG. V.21 : Confirmer l'ajout du service

Après la saisi des informations de service par API ou sans API le système va extraire les mots-clés à partir de description de service entrée, ces mots-clés sont stocker dans la propriété SLATokens de service.

Une étape d'update sera fait après chaque enregistrement d'un nouveau service. Le système va récupérer tous les FF qui ont des services dans notre BD, ainsi il récupère toutes les SLATokens des services après il cherche les mots-clés qui se répètent le plus dans notre SLATokens. Ces mots-clés doivent représenter les FF dans notre ontologie comme le montre la figure (V.22). Donc on va mettre à jour notre BD ainsi que les mots-clés qui représentent les FF dans notre ontologie on doit les enregistrer ces mots-clés dans le dictionnaire Cloud avec leurs synonymes pour après faciliter la tâche de découverte montré dans la figure (V.23).

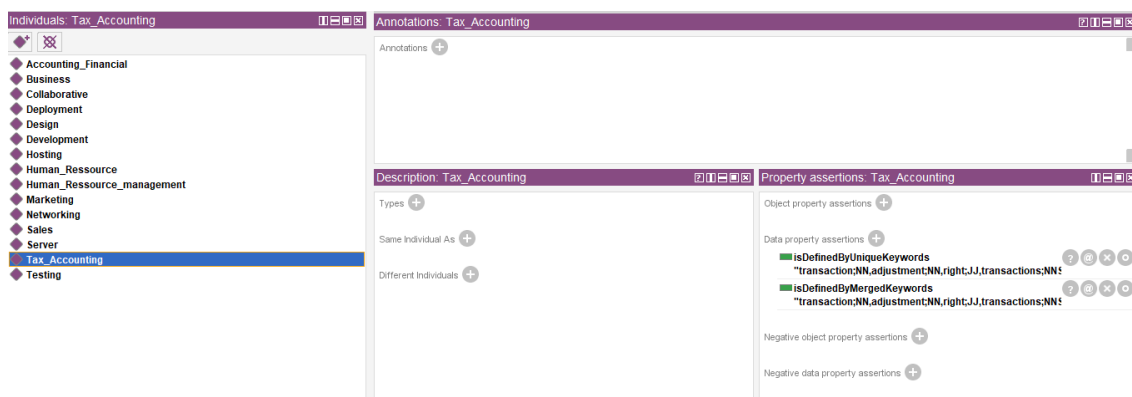


FIG. V.22 : Exemple de FF avec leur Unique et marged Keywords

```

1 [{"media;NN":
2 {"Concept":["tunica_media","media","media_wall","middle_coat","tunica_media_vasorum"]
3 "Entity":["rising","25_years_in_concert","atlantis","cosmic_wheels"]
4 "CloudDomainName":"StreamingAndMultimedia"}}
5 {"number;NN":
6 {"Concept":["number","keep_down","act","routine","turn","bit","grammatical_number","classifiers_with_number_morphology","distributive_plural","
7 "Entity":["number","sports_graphic_number","numbers","number[s]","what_to_do_after_you_hit_return","what_to_do_after_you_hit_return_or_p.c.c."
8 "CloudDomainName":"Tax_Accounting"}}
9 {"transaction;NN":
10 {"Concept":["trade","commercial_transaction","trading","transaction","buying","commercial_law","market_trade","market_trading","medieval_trade"
11 "Entity":["transaction"]
12 "CloudDomainName":"Tax_Accounting"}}
13 {"numbers;NNS":
14 {"Concept":["numbers","numbers_season_4","list_of_numb3rs_episodes","list_of_numbers_episodes","numb3rs","numbers_pool","numbers_game","number
15 "Entity":["numbers","numbers.""i_will_learn_to_love_again","numb3rs","numbers_season_3","numbers_season_6","young_legionnaire","chapter_ver
16 "CloudDomainName":"Tax_Accounting"}},
17 {"adjustment;NN":
18 {"Concept":["adjustment","social_adjustment","personal_adjustment","psychological_adjustment","performance_tuning","tuning","tweaking","regist
19 "Entity":[]
20 "CloudDomainName":"Tax_Accounting"}},
21 {"transactions;NNS":
22 {"Concept":["minutes","proceedings","transactions","meeting_minutes","minutes_of_meeting"]
23 "Entity":[]
24 "CloudDomainName":"Tax_Accounting"}},
25 {"right;JJ":
26 {"Concept":["justly","right","good","ripe","right_hand_side","right_side","correct","properly","decently","decent","in_good_order","the_right_w
27 "Entity":["right","conservative_party_of_norway","høyre","hogre","hoyre","hogre","høyre.no","leader_of_the_conservative_pa
28 "CloudDomainName":"Tax_Accounting"}]}

```

FIG. V.23 : Exemple de Dictionnaire Cloud

La figure (V.24) fournis un extrait du modèle BDR développé des NFF couverts dans notre ontologie.

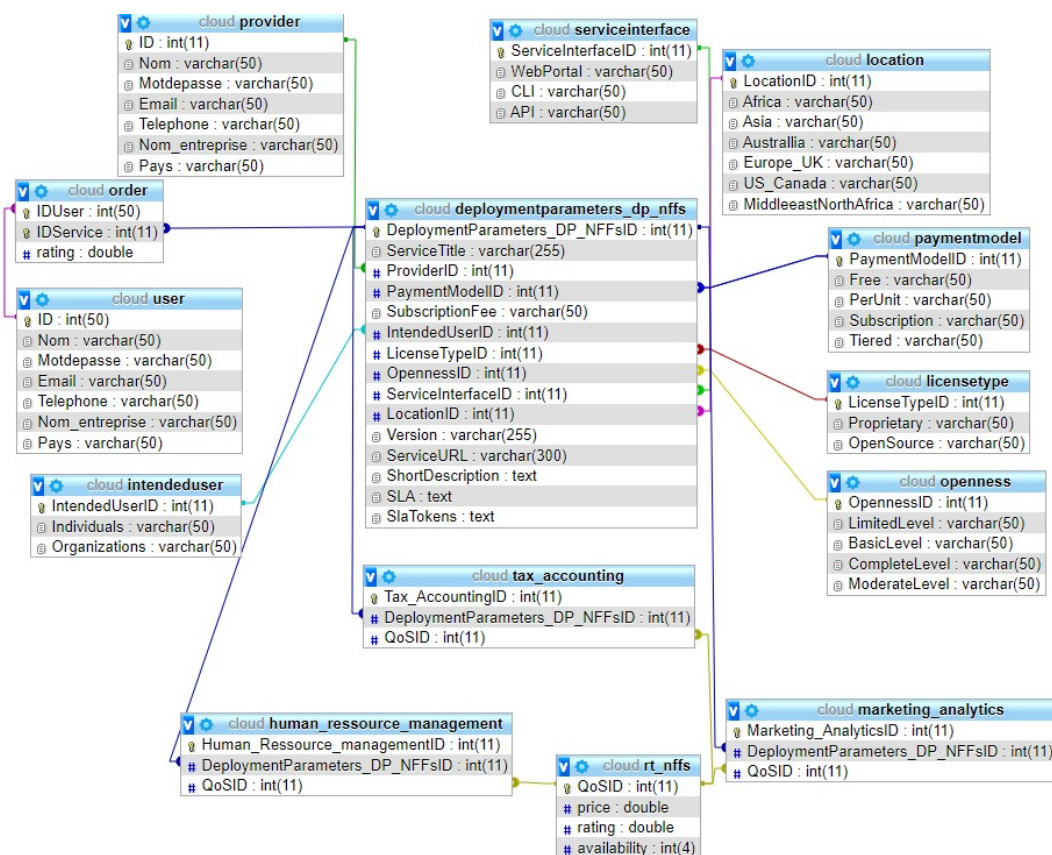


FIG. V.24 : Un extrait de notre BDR

V.4.2 Espace Utilisateur

Lorsque l'utilisateur se connecte, il est redirigé vers l'interface "Search for service" il écrit sa requête en langage naturel anglais précisant le service désiré comme illustré dans la figure (V.26), ensuite il clique sur le bouton search.

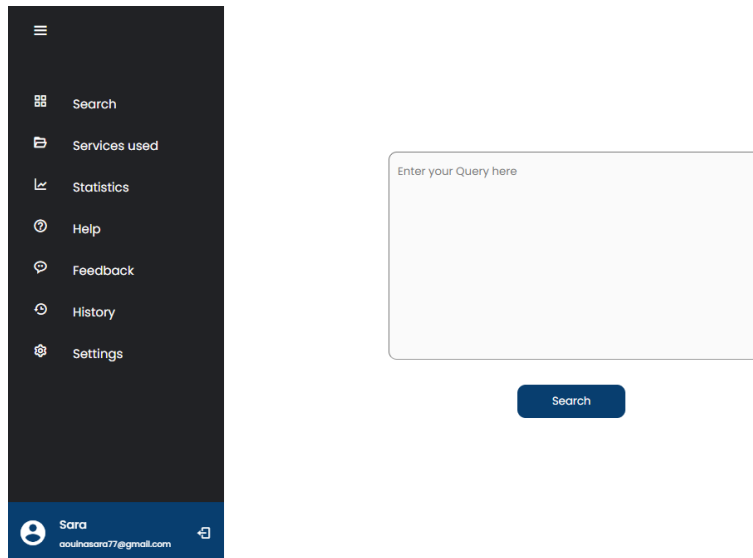


FIG. V.25 : Interface User

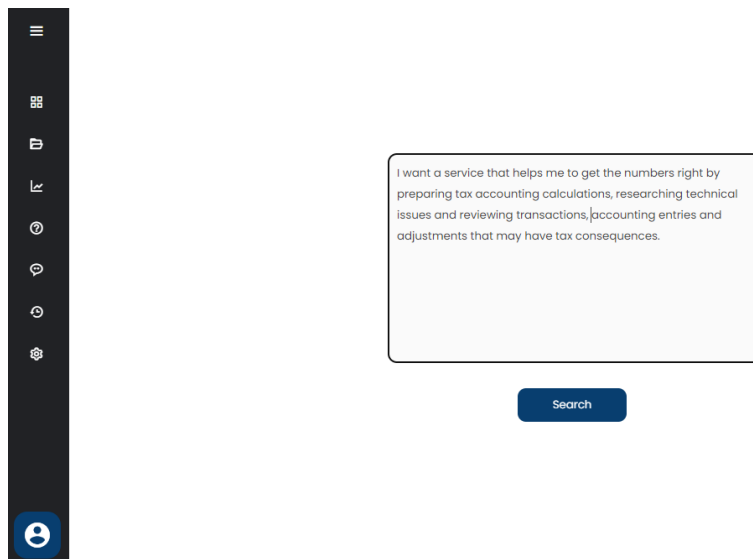
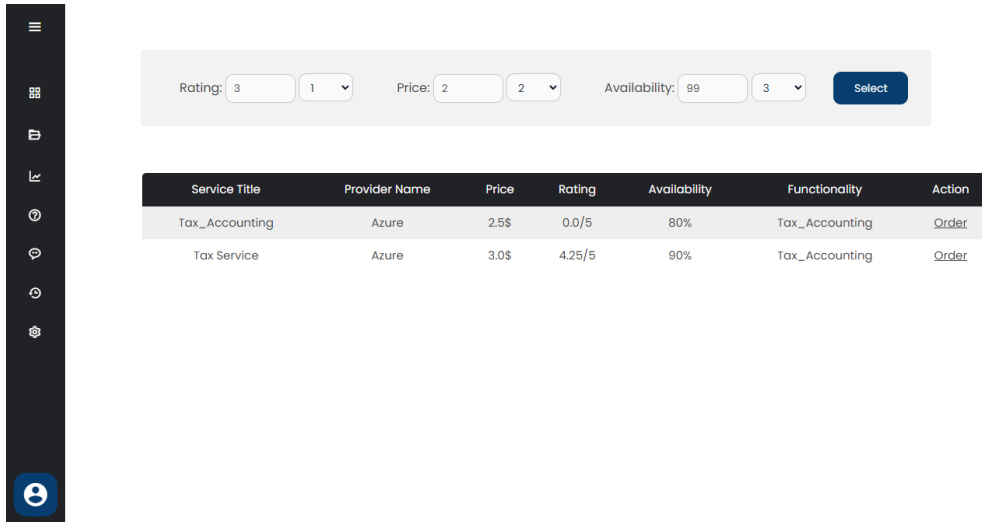


FIG. V.26 : Interface "Search for service"

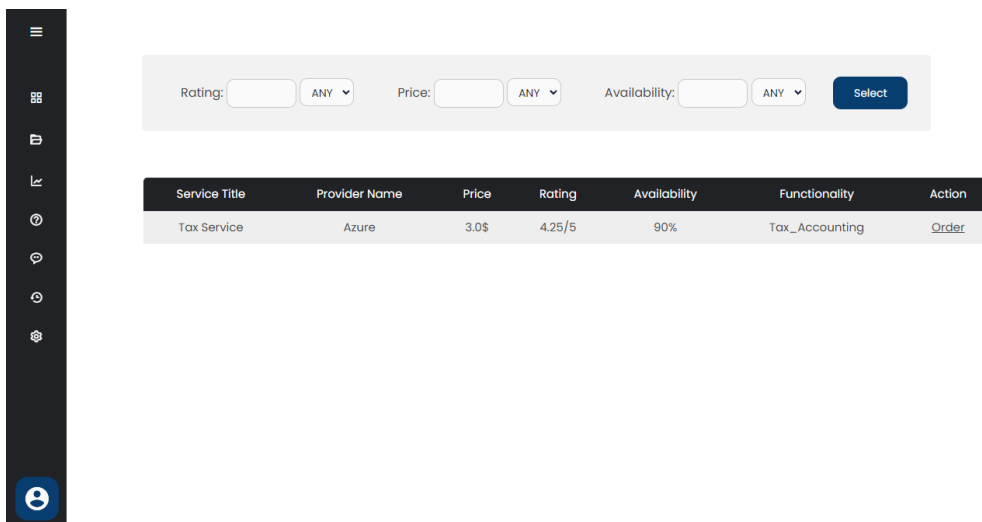
Lors du traitement de la requête, le système va extraire les expressions clés à partir de la requête entrée, après il calcule la similarité entre les mots-clés de requête et les mots uniques de chaque FF dans l'ontologie, une liste des services similaires dans la similarité

entre les mots-clés de la requête et de concept ontologique qui dépassent 60% est apparus aux utilisateurs comme illustré dans la figure (V.27). À ce moment, l'utilisateur doit choisir le meilleur service qui existe entre eux, il doit donc remplir le poids de chaque critère de qualité de service et les valeurs requises en fonction de son besoin, comme le montre la figure (V.28).



Service Title	Provider Name	Price	Rating	Availability	Functionality	Action
Tax_Accounting	Azure	2.5\$	0.0/5	80%	Tax_Accounting	Order
Tax Service	Azure	3.0\$	4.25/5	90%	Tax_Accounting	Order

FIG. V.27 : Résultat de découverte



Service Title	Provider Name	Price	Rating	Availability	Functionality	Action
Tax Service	Azure	3.0\$	4.25/5	90%	Tax_Accounting	Order

FIG. V.28 : Interface de sélection

Si l'utilisateur souhaite voir plus de détails sur un service rendu dans l'étape de découverte ou de sélection, il appuie sur "Order" pour consulter le service comme indiqué sur la figure (V.29).

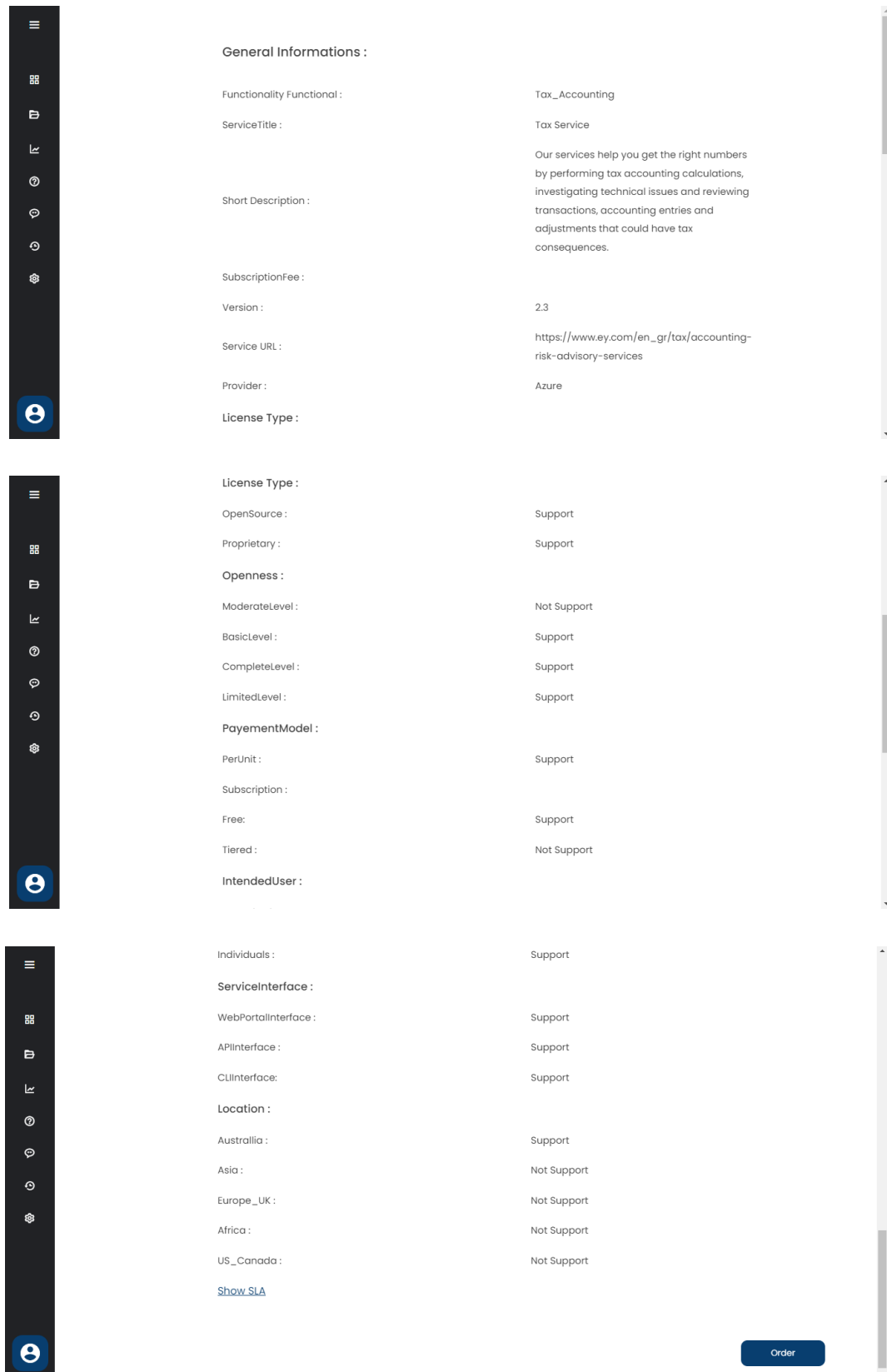


FIG. V.29 : Interface Paramètres de service

Si le client clique sur "Show SLA", il sera redirigé vers l'interface qui affiche le contrat SLA et ses détails tels que le service concerné, leur critère de qualité de service, la trafication, les

garanties et les pénalités.

En ce qui concerne la trafication, il s'agit du prix de service, tandis que le Broker bénéficie de 10% de la somme de tarifs, ce qui signifie que la trafication totale devient de la somme du tarif de service plus la part du Broker.

Les figures suivantes montrent le détail du SLA d'un service.



FIG. V.30 : Interface de SLA

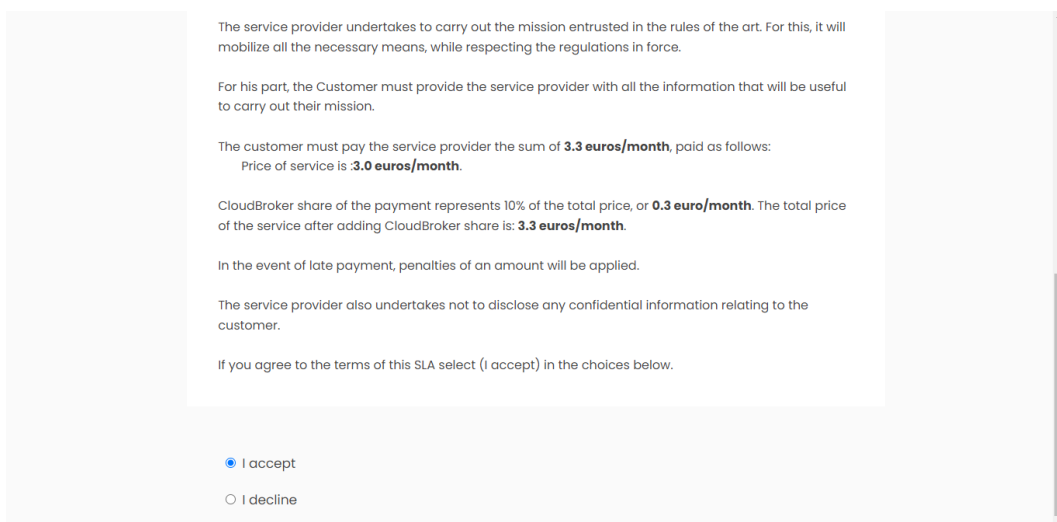
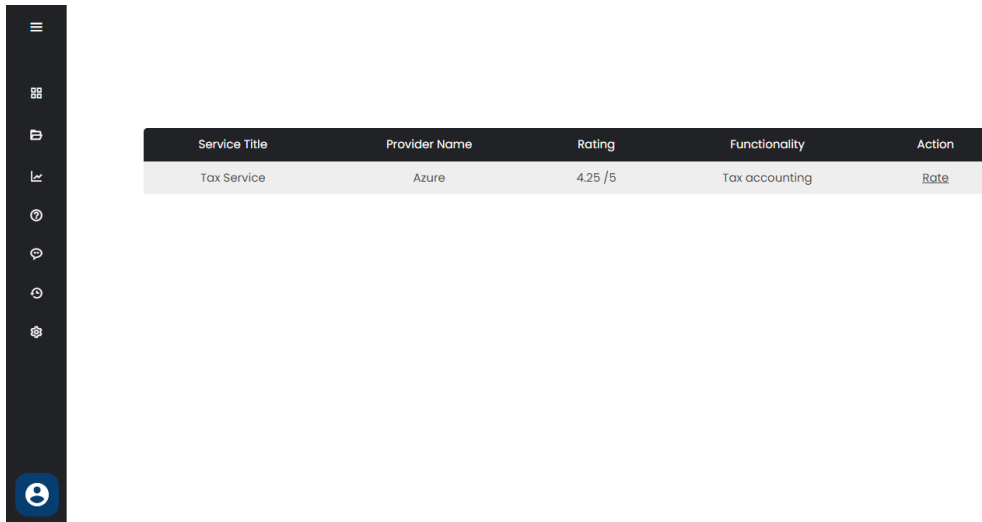


FIG. V.31 : Interface de SLA

Un client peut consulter la liste de ses services, comme le montre la figure suivante :

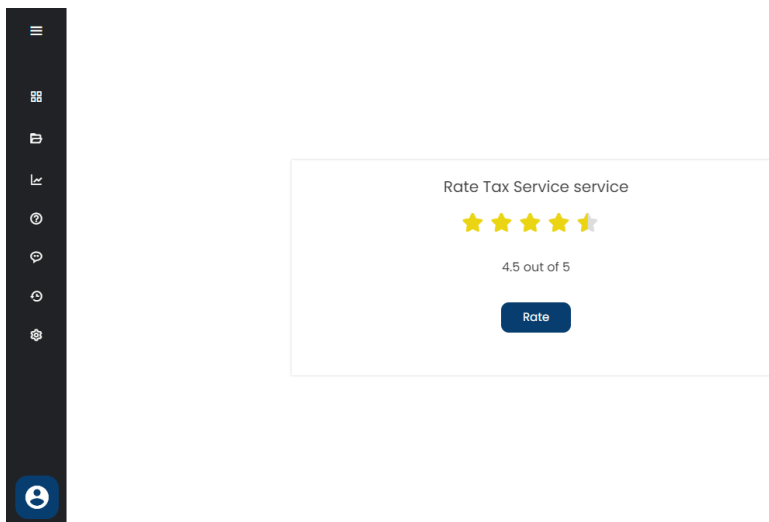


The screenshot shows a vertical sidebar on the left with various icons, including a user profile icon at the bottom. The main content area displays a table with the following data:

Service Title	Provider Name	Rating	Functionality	Action
Tax Service	Azure	4.25 /5	Tax accounting	Rate

FIG. V.32 : Interface de services utilisées

Un Client peut évaluer son service, comme indiqué dans la figure suivante :



The screenshot shows a vertical sidebar on the left with various icons, including a user profile icon at the bottom. The main content area displays a rating form for the 'Tax Service' service. The form includes the text 'Rate Tax Service service', a star rating of 4.5 out of 5 (represented by four full stars and one half star), and a 'Rate' button.

FIG. V.33 : Interface de réputation de service

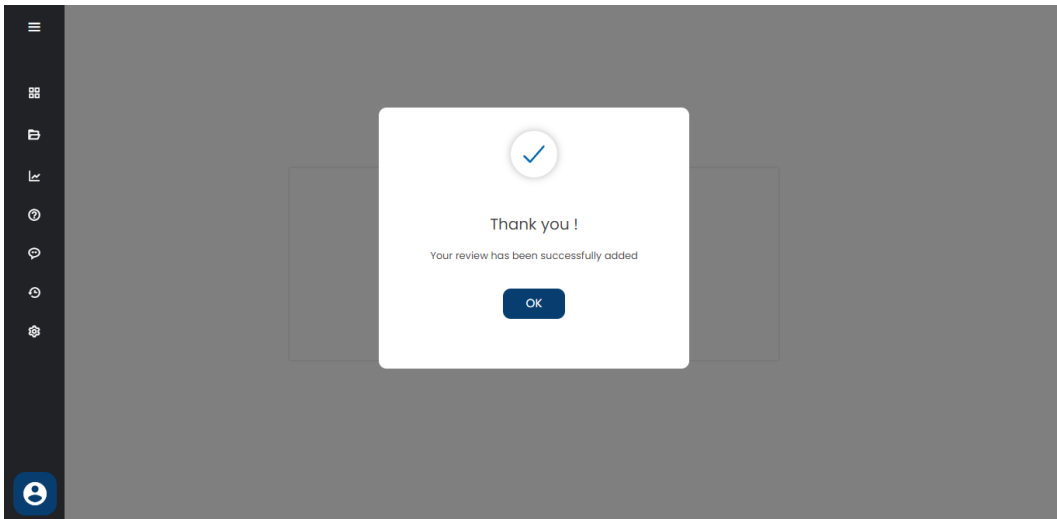


FIG. V.34 : Confirmation de rating

V.5 Évaluation de la solution par rapport à l'état de l'art

Pour évaluer notre solution, nous avons décidé de la comparer avec les solutions que nous avons étudiées dans le chapitre III selon les critères d'évaluation utilisés dans l'étude comparative. Cette évaluation est présentée dans le tableau ci-dessous :

<i>Solution</i>	Notre Solution
<i>Critère</i>	
C1	Non traité
C2	Traité
C3	Traité
C4	Traité
C5	Langage naturel
C6	Langage naturel - API
C7	Description - Découverte - sélection
C8	Détaillée
C9	Rapide
C10	Traité
C11	BD
C12	IaaS, PaaS et SaaS
C13	Centralisée + SMA + Ontop
C14	Domaine + Description

TAB. V.1 : Évaluation de notre solution

V.6 Conclusion

Au fil de ce dernier chapitre, nous avons montré l'efficacité de l'approche proposée en montrant les différents outils de développement et les résultats obtenus. En premier lieu, nous avons introduit quelques définitions des outils utilisés. Ensuite, nous avons illustré les interfaces principales de l'application par des captures écran. Nous avons terminé l'implémentation de notre application en respectant la conception élaborée. Enfin, une comparaison de l'approche proposée avec les approches décrites dans l'état de l'art est présentée.

Conclusion et perspectives

Le Cloud Computing est une grande évolution technologique dans le domaine informatique. C'est un paradigme dans lequel les entreprises peuvent stocker leurs données à distance et accéder aux services partout, là où ils en ont besoin et à moindres coûts.

Nous avons fait notre travail suivant la démarche Extreme Programming, nous avons commencé par étudier les concepts de base du Cloud Computing ses caractéristiques, avantages et inconvénients. Et puis nous avons étudié le problème de description, découverte et sélection de service dans le Cloud Computing. Plusieurs chercheurs ont proposé des modèles et des architectures pour obtenir de bons résultats qui satisfaisaient les utilisateurs, c'est ce qu'on a vu dans l'état de l'art de ce travail. L'objectif principal de la recherche menée dans ce mémoire réside dans la proposition d'une solution Broker pour la découverte des services Cloud.

Finalement nous avons mentionné les méthodes appliquées et les outils utilisés dans notre modèle qui est basé sur les travaux effectués par les recherches mentionnés précédemment, nous avons proposé une ontologie, qui a pour rôle de donner une description uniforme des services Cloud et pour ajouter un aspect sémantique à la solution. Ensuite, nous avons proposé une recherche basée sur les propriétés fonctionnelles, dont le but d'offrir un ensemble de services ayant les mêmes propriétés fonctionnelles du service désiré. De plus, nous avons ajouter une sélection pour sélectionner le meilleur service approprie aux exigences d'utilisateurs. Nous avons montré l'efficacité de l'approche proposée à travers les résultats obtenus par l'implémentation de l'approche proposée.

Ce travail a été une chance pour nous pour découvrir un nouveau visage de l'informatique moderne ce qui nous a permis d'acquérir une expérience significative dans le domaine Cloud qui est assez vaste et complexe, et d'approfondir nos connaissances concernant plusieurs aspects et phénomènes.

Ce travail a plusieurs perspectives potentielles, notamment la prise en charge de requêtes plus complexes d'utilisateurs et de fournisseurs, prise en considération des API des fournisseurs dans plusieurs langages tels que XML, etc. Implémenter la notion de composition des services, ajouter plusieurs domaines et fonctionnalités à notre ontologie, introduire d'autres valeurs de qualité de service dans le processus de sélection de services dans le but d'améliorer les résultats. De plus, nous aimerions pouvoir tester le travail de Broker en utilisant des services réels pour traiter l'aspect de l'invocation des services et voir les performances de notre solution en action.

Bibliographie

- [1] G. PLOUIN, *Cloud computing sécurité, gouvernance du SI hybride et panorama du marché*, Fr, 4e édition. France : DUNOD, 2016.
- [2] M. M. AL-SAYED, H. A. HASSAN et F. A. OMARA, «An intelligent cloud service discovery framework,» en, *Future Generation Computer Systems*, t. 106, p. 438-466, mai 2020. adresse : <https://linkinghub.elsevier.com/retrieve/pii/S0167739X19322630> (visité le 07/05/2022).
- [3] G. ABBAS, A. MEHMOOD, J. LLORET, M. S. RAZA et M. IBRAHIM, «FIPA-based reference architecture for efficient discovery and selection of appropriate cloud service using cloud ontology : FIPA-based reference architecture for cloud service,» en, *International Journal of Communication Systems*, t. 33, n° 14, e4504, sept. 2020. adresse : <https://onlinelibrary.wiley.com/doi/10.1002/dac.4504> (visité le 11/05/2022).
- [4] V. RAJARAMAN, «Cloud computing,» en, *Resonance – Journal of Science Education*, t. 19, n° 3, p. 242-258, mar. 2014. adresse : <https://doi.org/10.1007/s12045-014-0030-1> (visité le 11/05/2022).
- [5] D. E. Y. SARNA, *Implementing and Developing Cloud Computing Applications*, En, 1er édition. New York : Auerbach Publications, nov. 2010.
- [6] R. BUYYA, J. BROBERG et A. GOSCINSKI, *Cloud Computing Principles and Paradigms*, En, 1st edition. Wiley–Blackwell, mar. 2011.
- [7] D.-N. LE, R. KUMAR, G. N. NGUYEN et J. M. CHATTERJEE, *Cloud Computing and Virtualization*, En, 1er édition. USA : John Wiley-Scrivener, mar. 2018.
- [9] C. N. HOEFER et G. KARAGIANNIS, «Taxonomy of cloud computing services,» en, in *2010 IEEE Globecom Workshops*, Miami, FL, USA : IEEE, déc. 2010, p. 1345-1350. adresse : <http://ieeexplore.ieee.org/document/5700157/> (visité le 30/04/2022).

- [10] M. J. KAVIS, *Architecting the Cloud : Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS)*. Anglais. Hoboken, New Jersey : John Wiley & Sons, mar. 2014.
- [11] A. RASHID et A. CHATURVEDI, «Cloud Computing Characteristics and Services : A Brief Review,» en, *International Journal of Computer Sciences and Engineering*, t. 7, n° 2, p. 421-426, 2019.
- [12] S. K. SOOD, «A combined approach to ensure data security in cloud computing,» en, *Journal of Network and Computer Applications*, t. 35, n° 6, p. 1831-1838, nov. 2012. adresse : <https://linkinghub.elsevier.com/retrieve/pii/S1084804512001592> (visité le 19/05/2022).
- [14] P. COSTA, M. MIGLIAVACCA, P. PIETZUCH et A. L. WOLF, «NaaS : Network-as-a-Service in the Cloud,» in *2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE 12)*, San Jose, CA : USENIX Association, avr. 2012. adresse : <https://www.usenix.org/conference/hot-ice12/workshop-program/presentation/costa> (visité le 12/05/2022).
- [16] S. M. HABIB, S. HAUKE, S. RIES et M. MÜHLHÄUSER, «Trust as a facilitator in cloud computing : a survey,» *Journal of Cloud Computing : Advances, Systems and Applications*, t. 1, n° 1, p. 1-18, août 2012. adresse : <https://doi.org/10.1186/2192-113X-1-19> (visité le 19/05/2022).
- [17] B. R. KANDUKURI, R. P. V. et A. RAKSHIT, «Cloud Security Issues,» en, in *2009 IEEE International Conference on Services Computing*, Bangalore, India : IEEE, 2009, p. 517-520. adresse : <http://ieeexplore.ieee.org/document/5283911/> (visité le 23/05/2022).
- [18] G. J. MIROBI et L. AROCKIAM, «Service Level Agreement in cloud computing : An overview,» en, in *2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, Kumaracoil, India : IEEE, déc. 2015, p. 753-758. adresse : <https://ieeexplore.ieee.org/document/7475380> (visité le 21/04/2022).
- [20] J. W. RITTINGHOUSE et J. F. RANSOME, *Cloud Computing : Implementation, Management, and Security*, En, 1er édition. CRC Press, août 2009.

- [21] R. HILL, L. HIRSCH, P. LAKE et S. MOSHIRI, *Guide to Cloud Computing*, en, sér. Computer Communications and Networks. London : Springer London, 2013. adresse : <http://link.springer.com/10.1007/978-1-4471-4603-2> (visité le 12/06/2022).
- [23] J. S. HURWITZ, *Cloud computing for dummies*, English, 2nd édition. Indianapolis : John Wiley et Sons, sept. 2020.
- [26] A. T. VELTE, T. J. VELTE et R. C. ELSERPETER, *Cloud computing : a practical approach*, En, 1st edition. New York : McGraw-Hill Education, 2010.
- [27] C. WOLFHUGEL, L. BLOCH, A. SOUILLÉ, A. KOKOS et G. BILLOIS, *Sécurité informatique : pour les DSI, RSSI et administrateurs*, Fr, 5e éd. Paris : Eyrolles, oct. 2016.
- [29] S. GHAZOUANI et Y. SLIMANI, «A survey on cloud service description,» en, *Journal of Network and Computer Applications*, t. 91, p. 61-74, août 2017. adresse : <https://linkinghub.elsevier.com/retrieve/pii/S1084804517301613> (visité le 24/04/2022).
- [30] F. NAWAZ, A. MOHSIN et N. K. JANJUA, «Service description languages in cloud computing : state-of-the-art and research issues,» en, *Service Oriented Computing and Applications*, t. 13, n° 2, p. 109-125, juin 2019. adresse : <http://link.springer.com/10.1007/s11761-019-00263-z> (visité le 24/04/2022).
- [31] F. MOHAMMED, A. M. ALI, A. S. A.-M. AL-GHAMDI, F. ALSOLAMI, S. M. SHAMSUDDIN et F. E. EASSA, «Cloud Computing Services : Taxonomy of Discovery Approaches and Extraction Solutions,» en, *Symmetry*, t. 12, n° 8, p. 1354, août 2020. adresse : <https://www.mdpi.com/2073-8994/12/8/1354> (visité le 25/04/2022).
- [32] S. KUYORO, A. OKOLIE, R. AKINDE et S. OKOLIE, «Quality of Service (Qos) Issues in Web Services,» en, *IJCSNS International Journal of Computer Science and Network Security*, t. 12, n° 1, p. 94-96, jan. 2012.
- [33] R. KARIM, C. DING et A. MIRI, «An End-to-End QoS Mapping Approach for Cloud Service Selection,» en, in *2013 IEEE Ninth World Congress on Services*, Santa Clara, CA, USA : IEEE, juin 2013, p. 341-348. adresse : <http://ieeexplore.ieee.org/document/6655719/> (visité le 25/04/2022).
- [34] M. RAJESWARI, G. SAMBASIVAM, N. BALAJI, M. SALEEM BASHA, T. VENGATTARAMAN et P. DHAVACHELVAN, «Appraisal and analysis on various web service composition approaches based on QoS factors,» en, *Journal of King Saud University - Computer and*

- Information Sciences*, t. 26, n° 1, p. 143-152, jan. 2014. adresse : <https://linkinghub.elsevier.com/retrieve/pii/S1319157813000281> (visité le 26/04/2022).
- [35] S. K. GARG, S. VERSTEEG et R. BUYYA, «A framework for ranking of cloud computing services,» en, *Future Generation Computer Systems*, t. 29, n° 4, p. 1012-1023, juin 2013. adresse : <https://linkinghub.elsevier.com/retrieve/pii/S0167739X12001422> (visité le 25/04/2022).
- [36] J. CARDOSO, A. BARROS, N. MAY et U. KYLAU, «Towards a Unified Service Description Language for the Internet of Services : Requirements and First Developments,» en, in *2010 IEEE International Conference on Services Computing*, Miami, FL, USA : IEEE, juil. 2010, p. 602-609. adresse : <http://ieeexplore.ieee.org/document/5557283/> (visité le 03/04/2022).
- [37] D. MARTIN, M. BURSTEIN, D. MCDERMOTT, S. MCILRAITH, M. PAOLUCCI, K. SYCARA, D. L. MCGUINNESS, E. SIRIN et N. SRINIVASAN, «Bringing Semantics to Web Services with OWL-S,» en, *World Wide Web*, t. 10, n° 3, p. 243-277, sept. 2007. adresse : <https://link.springer.com/10.1007/s11280-007-0033-x> (visité le 03/03/2022).
- [39] A. M. ALKALBANI et F. K. HUSSAIN, «A comparative study and future research directions in cloud service discovery,» en, in *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*, Hefei, China : IEEE, juin 2016, p. 1049-1056. adresse : <http://ieeexplore.ieee.org/document/7603737/> (visité le 02/03/2022).
- [40] M. PARHI, B. K. PATTANAYAK et M. R. PATRA, «A Multi-agent-Based Framework for Cloud Service Description and Discovery Using Ontology,» en, in *Intelligent Computing, Communication and Devices*, L. C. JAIN, S. PATNAIK et N. ICHALKARANJE, éd., t. 308, Series Title : Advances in Intelligent Systems and Computing, New Delhi : Springer India, 2015, p. 337-348. adresse : http://link.springer.com/10.1007/978-81-322-2012-1_35 (visité le 03/03/2022).
- [41] Z. u. REHMAN, F. K. HUSSAIN et O. K. HUSSAIN, «Towards Multi-criteria Cloud Service Selection,» in *2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, Seoul, Korea (South) : IEEE, juin 2011, p. 44-48. adresse : <http://ieeexplore.ieee.org/document/5976164/> (visité le 21/05/2022).

- [42] C. BUSSLER, D. FENSEL et A. MAEDCHE, «A conceptual architecture for semantic web enabled web services,» en, *ACM SIGMOD Record*, t. 31, n° 4, p. 24-29, déc. 2002. adresse : <https://dl.acm.org/doi/10.1145/637411.637415> (visité le 21/06/2022).
- [43] G. GARDARIN, *XML : des bases de données aux services Web*, 2^e éd., sér. InfoPro (Paris). Dunod, 2002. adresse : <https://books.google.dz/books?id=UNWrAAAACAAJ> (visité le 21/06/2022).
- [44] M. REKIK, K. BOUKADI et H. BEN-ABDALLAH, «Cloud Description Ontology for Service Discovery and Selection :» en, in *Proceedings of the 10th International Conference on Software Engineering and Applications*, Colmar, Alsace, France : SCITEPRESS - Science, 2015, p. 26-36. adresse : <http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005556400260036> (visité le 03/03/2022).
- [45] J. KANG et K. M. SIM, «Ontology-enhanced agent-based cloud service discovery,» en, *International Journal of Cloud Computing*, t. 5, n° 1/2, p. 144-171, 2016. adresse : <http://www.inderscience.com/link.php?id=75125> (visité le 28/05/2022).
- [46] KWANG MONG SIM, «Agent-Based Cloud Computing,» en, *IEEE Transactions on Services Computing*, t. 5, n° 4, p. 564-577, 2012. adresse : <http://ieeexplore.ieee.org/document/6042853/> (visité le 30/05/2022).
- [47] T. HAN et K. M. SIM, «An Ontology-enhanced Cloud Service Discovery System,» en, *Lecture Notes in Engineering and Computer Science*, t. 2180, p. 7, 2010.
- [48] S. HASAN et V. VALLI KUMARI, «Generic-distributed framework for cloud services marketplace based on unified ontology,» en, *Journal of Advanced Research*, t. 8, n° 6, p. 569-576, nov. 2017. adresse : <https://linkinghub.elsevier.com/retrieve/pii/S2090123217300930> (visité le 30/05/2022).
- [49] A. ALFAZI, Q. Z. SHENG, A. BABAR, W. RUAN et Y. QIN, «Toward Unified Cloud Service Discovery for Enhanced Service Identification,» en, in *Service Research and Innovation*, A. BEHESHTI, M. HASHMI, H. DONG et W. E. ZHANG, éd., t. 234, Series Title : Lecture Notes in Business Information Processing, Cham : Springer International Publishing, 2018, p. 149-163. adresse : http://link.springer.com/10.1007/978-3-319-76587-7_10 (visité le 30/05/2022).

- [50] M. M. AL-SAYED, H. A. HASSAN et F. A. OMARA, «Towards evaluation of cloud ontologies,» en, *Journal of Parallel and Distributed Computing*, t. 126, p. 82-106, avr. 2019. adresse : <https://www.sciencedirect.com/science/article/pii/S0743731518306105> (visité le 09/05/2022).

Webographie

- [8] *IaaS, PaaS, SaaS – Présentation des modèles de services Cloud*, fr. adresse : <https://www.intel.com/content/www/fr/fr/cloud-computing/as-a-service.html> (visité le 21/05/2022).
- [13] *What Is Network as a Service (NaaS) ?* en. adresse : <https://www.cisco.com/c/en/us/solutions/enterprise-networks/network-as-service-naas.html> (visité le 21/03/2022).
- [15] *What is DBaaS (Database-as-a-Service) | IBM*. adresse : <https://www.ibm.com/cloud/learn/dbaas> (visité le 20/02/2022).
- [19] *Qu'est-ce qu'un centre de données ?* Fr, nov. 2020. adresse : <https://www.ibm.com/fr-fr/cloud/learn/data-centers> (visité le 22/04/2022).
- [22] a. thelma allen, *NIST Cloud Computing Program - NCCP*, English, text, Last Modified : 2019-07-09T11:34-04:00, nov. 2010. adresse : <https://www.nist.gov/programs-projects/nist-cloud-computing-program-nccp> (visité le 15/04/2022).
- [24] *Avantages du cloud computing*, fr-fr, fév. 2019. adresse : <https://www.ibm.com/fr-fr/cloud/learn/benefits-of-cloud-computing> (visité le 16/04/2022).
- [25] *AWS | Qu'est-ce que le cloud computing – Les avantages du cloud*, fr-FR. adresse : <https://aws.amazon.com/fr/what-is-cloud-computing/> (visité le 07/04/2022).
- [28] *Gartner Forecasts Worldwide Public Cloud Revenue to Grow 6.3% in 2020*, en. adresse : <https://www.gartner.com/en/newsroom/press-releases/2020-07-23-gartner-forecasts> (visité le 04/04/2022).
- [38] *Web Services Glossary*. adresse : <https://www.w3.org/TR/2004/NOTE-ws-gloss-2041/#defs> (visité le 30/04/2022).
- [51] *What is Eclipse ?* en. adresse : <https://www.educative.io/edpresso/what-is-eclipse> (visité le 25/04/2022).

Annexe A

Détails des travaux connexes

A.1 Solution de Rekik et al [44]

Les concepts de niveau supérieur de cette solution, sont indiqués dans la figure (A.1).

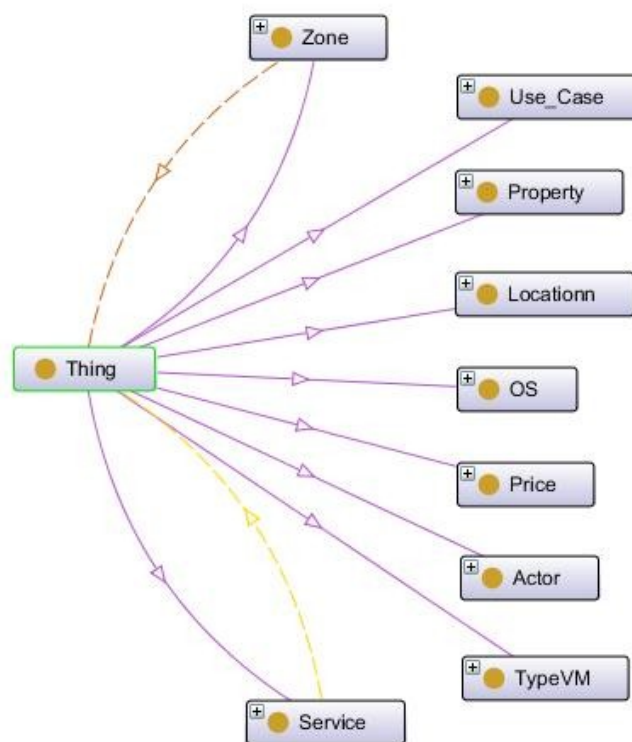


FIG. A.1 : Les concepts d'ontologie

1. Les classes :

- La classe Actor :

La classe Acteur présente les différents acteurs qui participer aux systèmes Cloud. Qui sont :

1. L'utilisateur / Consumer : Ils utilisent le moteur de recherche pour trouver leurs services souhaités, ce moteur interroge l'ontologie pour effectuer la tâche de la découverte.
2. Le fournisseur / Provider : Ils utilisent le vocabulaire (owl) de l'ontologie pour la description de leurs services.
3. Le Broker : Qui surveille, négocie, planifie, etc. les services Cloud.

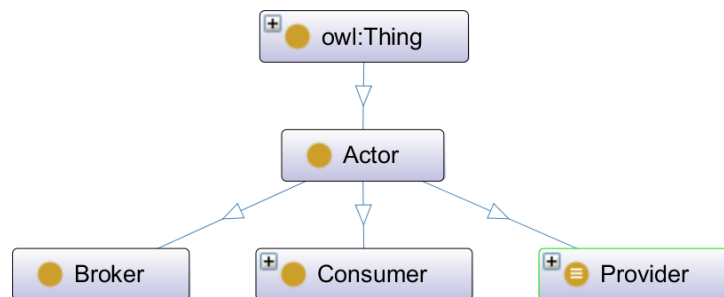


FIG. A.2 : la classe "Acteur".

- **La classe Service :**

Chaque fournisseur propose un service qui est en réalité une ressource virtualisée. C'est-à-dire qu'un fournisseur IaaS propose soit une machine virtuelle soit un espace de stockage, un fournisseur PaaS propose une plate-forme, tandis qu'un fournisseur SaaS propose un logiciel ou une application.

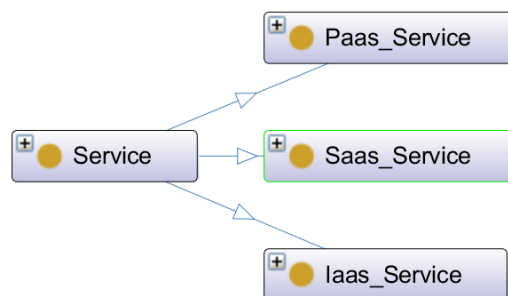


FIG. A.3 : La classe "Service"

1. Service SaaS :

La classe Software as a Service inclut les logiciels, les applications, etc.

2. Service PaaS :

La classe Platform as a Service inclut IDE (un environnement de développement intégré), API (une interface de programmation d'application), Application Server (un environnement où les applications peut s'exécuter), Web Server (serveur utilisé pour publier sites Web) et Data Base (une collection organisée de données) sous-classes. Toutes ces sous-classes citées font partie de la classe Platform.

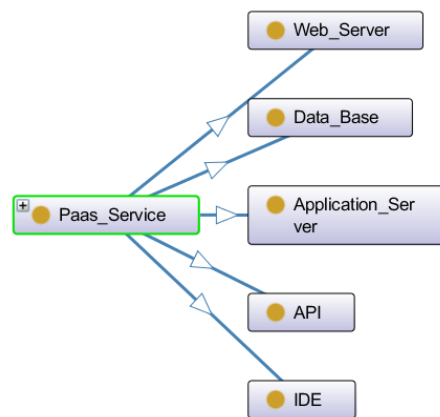


FIG. A.4 : La classe "PaaS"

3. Service IaaS :

La classe Infrastructure as a Service inclut :

- **Classe Virtual Machine** : VM a une localisation et un type. Selon le type, nous distinguons les caractéristiques de la VM.
- **Classe Storage Space** : Cette classe peut être fourni avec ou sans VM. Cette classe couvre Distributed-File-System (est un système de fichiers qui permet d'accéder aux fichiers sur les multiples hôtes du Cloud), Replicated-Relational-Database (est une base de données qui permet la copie de données à partir d'un base de données d'une ressource vers une base de données d'une autre) et les sous-classes Disk.

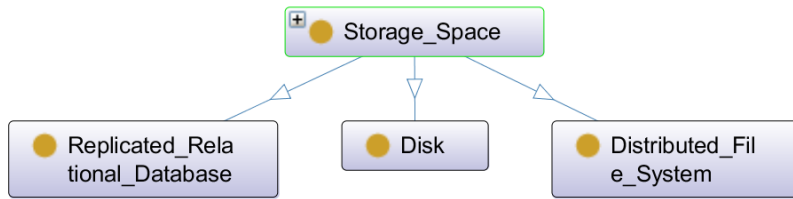


FIG. A.5 : La classe "Storage_Space"

- **La classe TypeVM** : Dans cette classe, nous décrivons les cinq caractéristiques majeures du type du VM, qui sont (Figure A.6) :

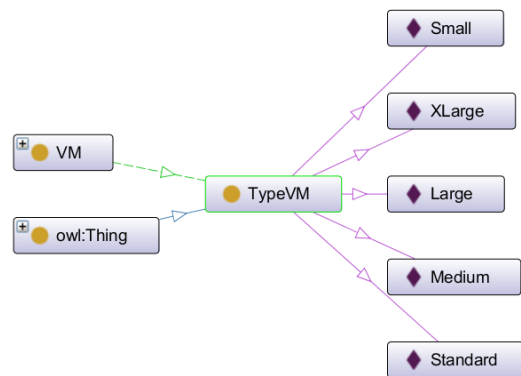


FIG. A.6 : La Classe "TypeVM"

- **La classe Use Case** : Chaque service de VM a un Use Case qui peut être soit :
 - Genral Use (Usage général).
 - Optimized Memory (Mémoire optimisée).
 - Optimized Calculation (Calcul optimisé).
- **La classe OS** : Cette classe représente le système d'exploitation d'une VM. Nous décrivons ci-dessous quelques Système d'exploitation de notre ontologie : Ubuntu, Ubuntu_Pro, CentOS, Windows_server, etc.
- **La classe Location** : Cette classe représente les location de chaque service dans le Cloud. Nous décrivons ci-dessous quelques location de notre ontologie : Asia_sud1, Europe_ouest1, US_Central, Asia_nord_est3, etc.
- **La classe Zone** : Chaque location de service possède une zone spécifique. Nous décrivons ci-dessous quelques zone de notre ontologie : Amérique_Central, Amérique_Ouest,

etc.

- **La classe Price** : Au niveau de cette classe, nous décrivons les différents type du paiement, qui sont (Figure A.7) :

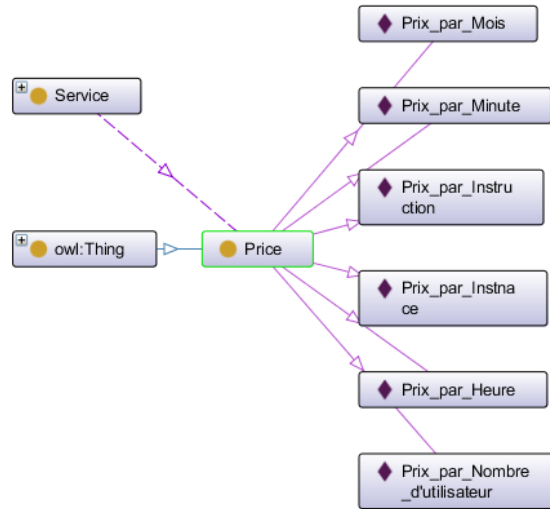


FIG. A.7 : La Classe "Price"

- **La classe Property** :

Les sous-classes Property contiennent tous les éléments nécessaires pour décrire les caractéristiques des ressources Cloud. Nous distinguons la classe de propriétés fonctionnelles et la classes de propriétés non fonctionnelles. Le premier définit la fonction de service (c'est-à-dire ce qu'un service est censé accomplir) tandis que le second spécifie la qualité ou les critères de service (QoS). Pour chaque service fourni (IaaS, PaaS et SaaS), nous définissons un ensemble de fonctionnalités et les propriétés non fonctionnelles.

1. La classe des propriétés fonctionnelles :

Cette dernière contient tous les éléments nécessaires pour la description de la fonctionnalité des services. Nous distinguons les trois sous classes :

- **Les propriétés fonctionnelles des services PaaS** : Cette classe dépend à la catégorie qu'un service PaaS appartient, c'est-à-dire les propriétés fonctionnelles d'un service PaaS sont incluses dans la liste des propriétés fonctionnelles de sa catégorie.
- **Les propriétés fonctionnelles des services IaaS** : Cette classe comprend les sous-classes Functional VM Property et Functional StorageSpace

Property.

- **Les propriétés fonctionnelles des services SaaS** : Décrit les caractéristiques d'un service software (SaaS).

2. La classe des propriétés non fonctionnelles :

Fondamentalement, les exigences non fonctionnelles concernent les qualités du service (QoS) qui croisent les fonctionnalités destinées aux utilisateurs, telles que la sécurité, la fiabilité, la disponibilité, etc. Dans cette classe, nous présentons les propriétés QoS et les propriétés non fonctionnelles des services (IaaS, PaaS et SaaS).

2. Les attributs d'ontologie :

La figure suivante (A.8) représente les différents attributs ou les propriétés de donnée (Data Properties) des différents concepts :

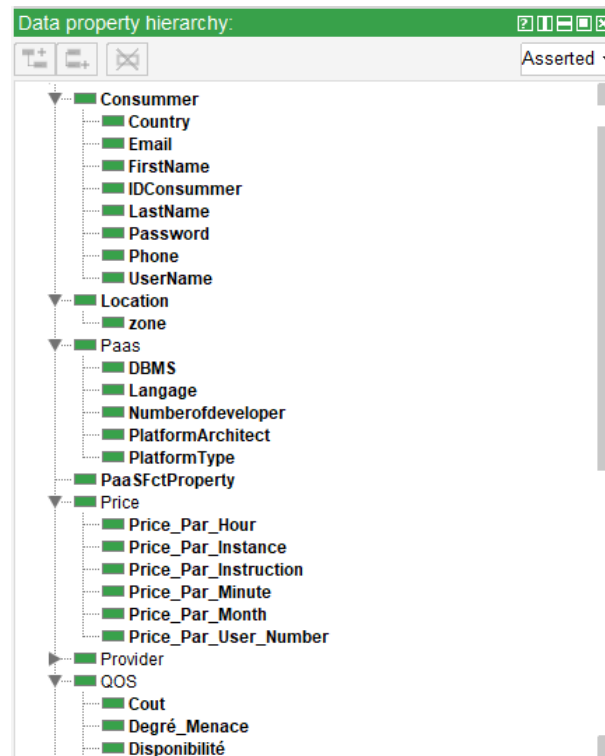


FIG. A.8 : Les attributs d'ontologie

3. Création du tableau des relations :

Dans le tableau ci-dessous (A.1), nous décrivons les différentes relations binaires existantes (Object Properties) entre les différents classes présentés :

Object properties	Domains	Ranges
Provide	Provide	Service
Has_zone	Service	Zone
Has_VMFctProperty	VM	VM Functional property
Has_Use_Case	VM	Use_Case
Has_TypeVM	VM	TypeVM
Has_StorageSpaceFctProperty	StorageSpace	StorageSpace_Functional_property
Has_SaaSfctProperty	SaaSService	SaaS_Functional_property
Has_PaaSfctProperty	PaasService	Paas_Functional_property
Has_QOS	Service	QOS
Has_OS	VM	OS
Has_Location	Service	Location
Has_ParentLocation	Zone	Location

TAB. A.1 : Les relations d'ontologie

A.2 Solution de Al-Sayed et al [2]

Cette solution contient 5 couches principales comme indiqué dans la figure suivante.

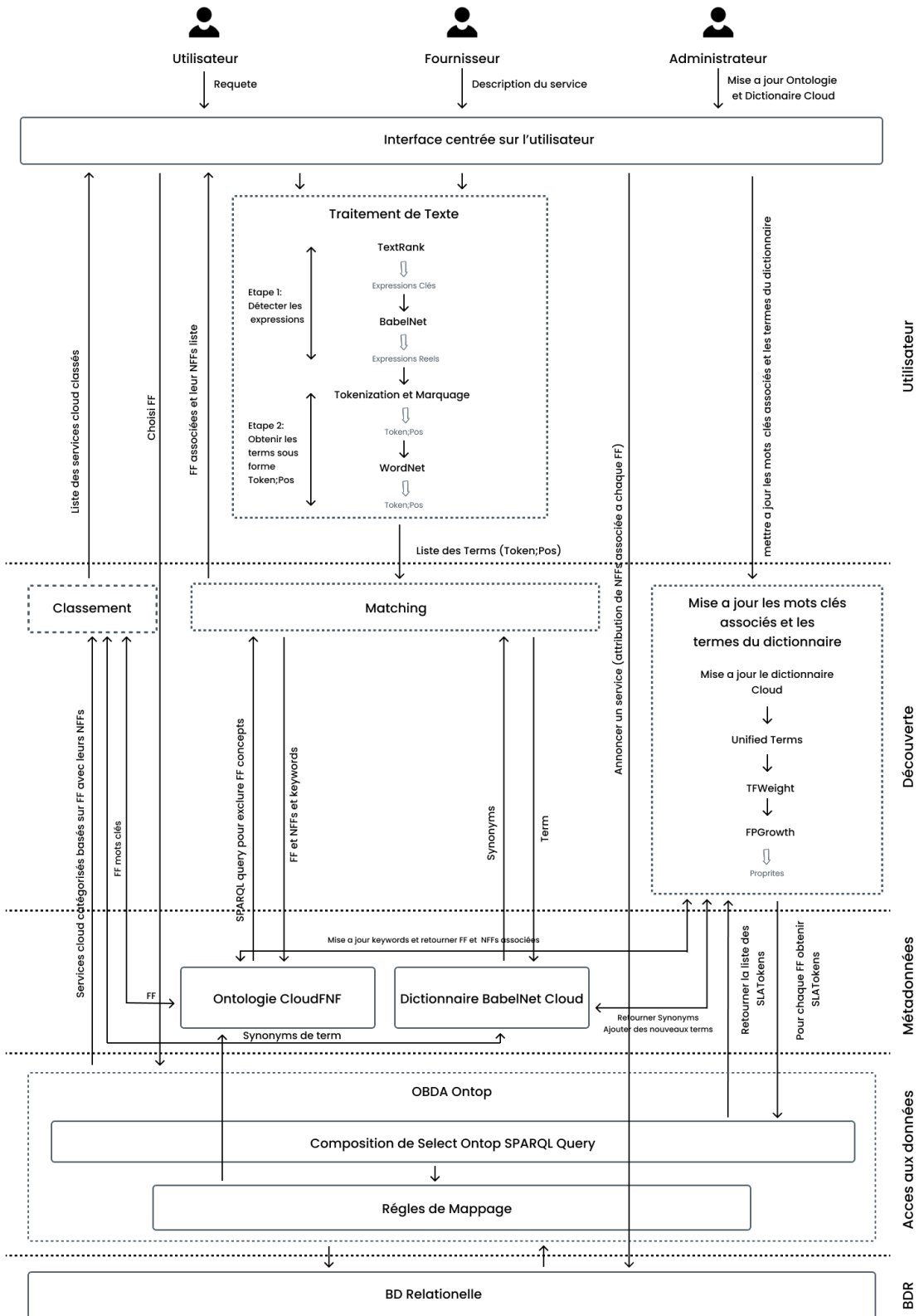


FIG. A.9 : Architecture de cadre de découvert de service Cloud selon la solution d'Al-Sayed et al [2]

1. La couche utilisateur

Trois parties peuvent communiquer via cette interface :

- **Utilisateur :**

Les utilisateurs utilisent l'interface pour découvrir les services Cloud. Ils commencent leur processus de découverte en composant des requêtes de découverte de texte détaillées en utilisant des termes familiers en langage naturel anglais sous une forme visualisée illustrée à la figure (A.10), ces termes en les fournissant à l'interface utilisateur comme entrée. Ces entrées sont envoyées au processus **Traitement de Texte**, ce processus contient deux étapes :

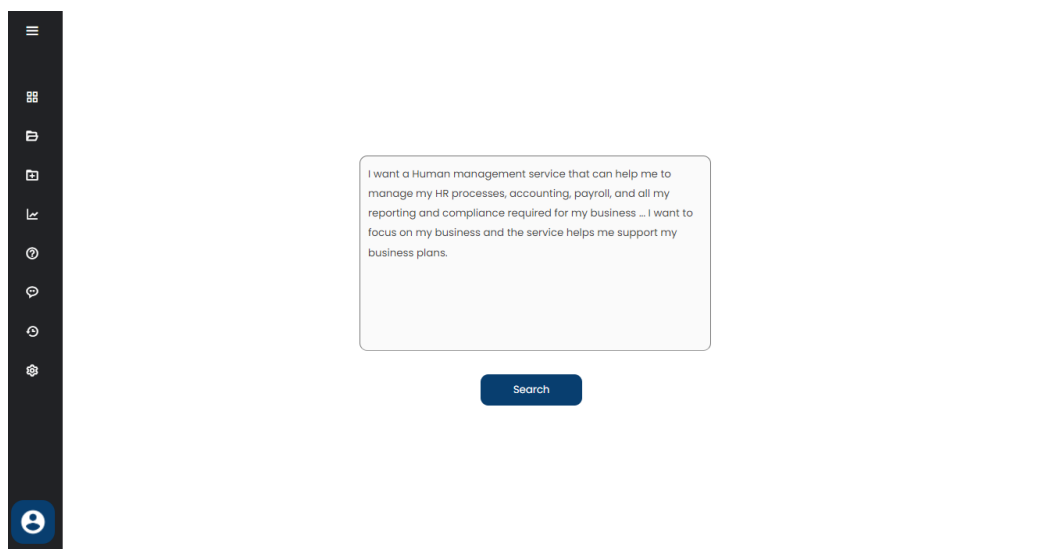


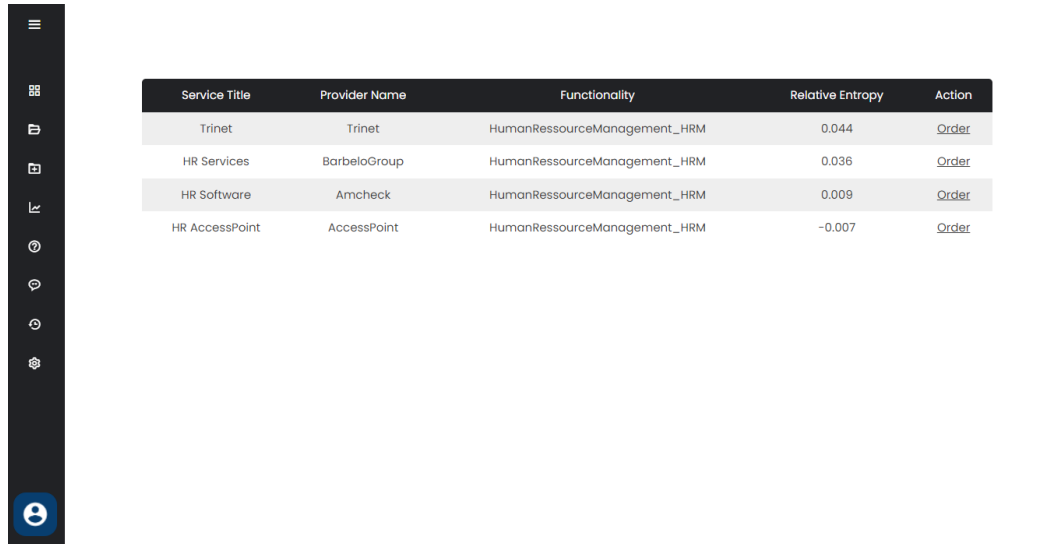
FIG. A.10 : Exemple de requête de texte utilisateur

1. Détecter les expressions clés : Dans cette étape nous avons l'algorithme **TextRank** pour détecter les expressions clés et ensuite faire passer ces expressions clés à **BabelNet** pour exclure les expressions fausses.

2. Obtenir les termes sous forme Token_Nom ;Type : Les expressions sorties de la première étape sont passées à **Tokenization et Marquage** des termes pour convertir les termes sous forme « Token_Nom ;Type », ensuite les termes seront convertis à sa base en utilisant **WordNet**, le résultat de ce processus va être sous forme « Token_Nom ;Type », ensuite on va passer au processus **Matching**.

Enfin, les services Cloud et leurs NFFs sont récupérés sur l'interface utilisateur, ou toutes les informations sur chaque service sont présentées en trois groupes : **les infor-**

mations générales, les **NFFs exclusifs** sur les FFs que le service peut fournir et le **degré de correspondance** par rapport à l'entrée de l'utilisateur sous une forme visualisée illustrée dans les figures (A.11).



Service Title	Provider Name	Functionality	Relative Entropy	Action
Trinet	Trinet	HumanResourceManagement_HRM	0.044	Order
HR Services	BarbeloGroup	HumanResourceManagement_HRM	0.036	Order
HR Software	Amcheck	HumanResourceManagement_HRM	0.009	Order
HR AccessPoint	AccessPoint	HumanResourceManagement_HRM	-0.007	Order

FIG. A.11 : Interface de services Cloud correspondante récupérée selon la requête d'utilisateur

- **Fournisseur :**

Les fournisseurs utilisent l'interface pour annoncer leurs services Cloud dans le référentiel développé. Ils saisissent les descriptions de leurs services dans le cadre de découverte développée afin de suggérer des FFs appropriés à classer. Ils commencent leur processus de publicité en attribuant les descriptions textuelles des services en utilisant des termes familiers en langage naturel anglais sous une forme visualisée, ces descriptions sont envoyées au processus **Traitement de Texte** qui suit les mêmes étapes comme la requête d'utilisateur, pour être converties sous la forme « Token_Nom ;Type » avant de passer au processus Matching.

Les fournisseurs peuvent sélectionner parmi FFs récupérés à partir de processus Matching et attribuer les valeurs nécessaires à leurs NFFs qui sont également présentés dans des **NFFs généraux** et **exclusifs** sous une forme visualisée illustrée dans les figures (A.12) (A.13).

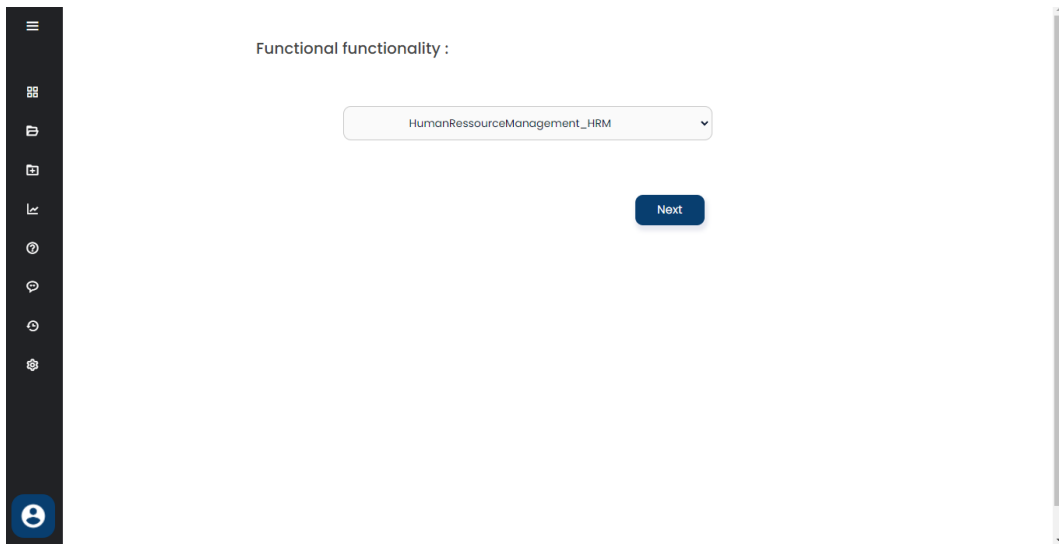


FIG. A.12 : FFs récupérés selon la description de fournisseur

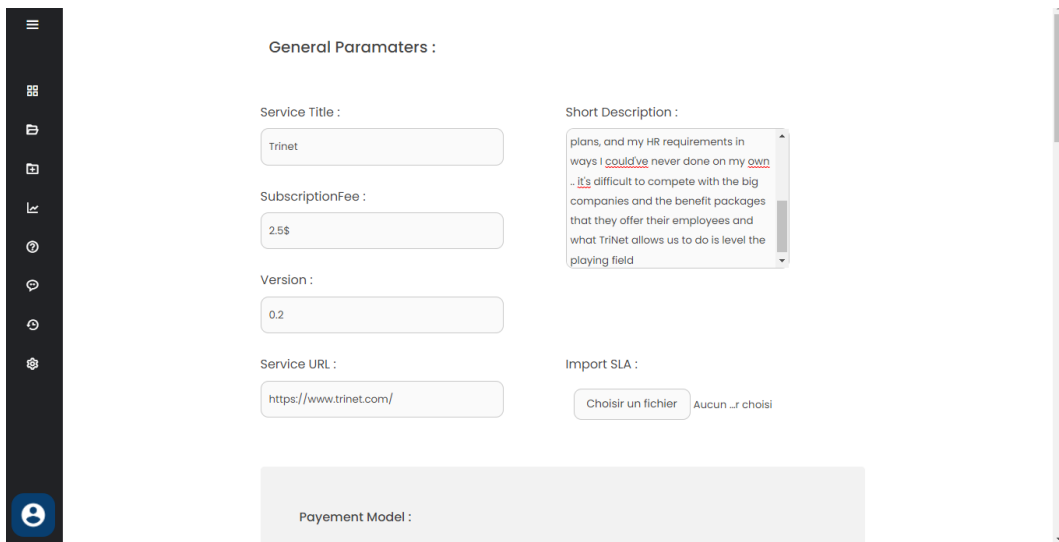


FIG. A.13 : Interface ajouter un service HRM

- **Administrateur** : Utilise l'interface pour maintenir les données à jour.

2. Métadonnées

La couche Métadonnées comprend deux composants principaux :

- **Ontologie CloudFNF** :

Cette Ontologie fournit une classification sémantique des services Cloud en fonction de leur FF en tant compte des interrelations entre ces FFs comme le montre la figure (A.15). De plus, les NFFs de ces services sont considérés dans cette ontologie pour

rendre le processus de classement des services, qui fournissent les mêmes FFs une opération efficace. Pour bien comprendre cette ontologie nous avons réalisé une ontologie CloudFNF similaire on utilisant l’outil Protégé.

Les concepts de l’ontologie référencée sont classés en cinq catégories (illustrées dans la figure A.14) :

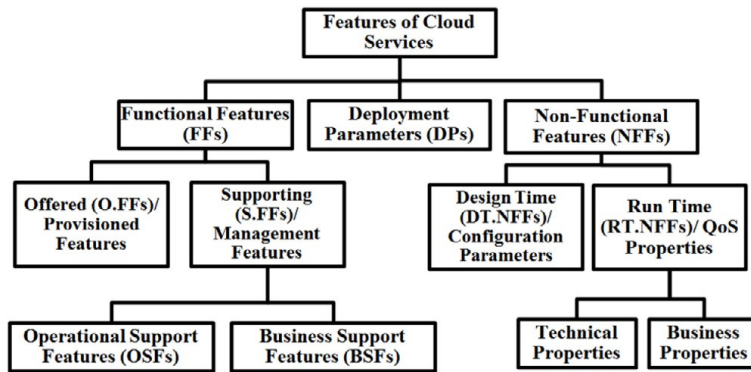


FIG. A.14 : Architecture générale des fonctionnalités des services selon l’ontologie CloudFNF

- **La catégorie des fonctionnalités fonctionnelles offertes (OFFs) :** Comprend les fonctionnalités publiées en tant que services individuellement ou dans des compositions pour répondre aux exigences fonctionnelles des utilisateurs du Cloud. Les services qui fournissent plus que FF sont classés sous chaque FF qu’ils fournissent.
- **La catégorie des fonctionnalités fonctionnelles de gestion (SFFs) :** Elle est similaire à la catégorie OFFs, mais elle inclut des fonctionnalités que d’autres fournisseurs de Cloud peuvent utiliser pour gérer efficacement leurs services et s’assurer que leurs OFFs sont fournis avec une haute qualité aux clients.
On distingue deux catégories de SFFs :
 - 1. Support métier :** Telle que l’allocation des ressources, la gestion des problèmes et la gestion des performances, etc.
 - 2. Support opérationnel :** Telle que la gestion des comptes clients et la gestion des commandes, etc.
- **La catégorie des fonctionnalités non fonctionnelles de déploiement (DeploymentParameters_DP_NFFs) :** Elle inclut les NFFs généraux, tels que le titre du service, l’URL du service et le nom du fournisseur, etc. pour chaque service Cloud.

- **La catégorie des fonctionnalités non fonctionnelles au moment de la conception ou configuration (DT.NFFs) :** Elle inclut des groupes de paramètres de configuration exclusifs associés à chaque FF. Chaque FF possède ses propres paramètres de configuration spécifique.
- **La catégorie des fonctionnalités non fonctionnelles au moment de l'exécution (RT.NFFs) :** Comprends un ensemble de paramètres communs pour faciliter l'évaluation de la qualité des services fournis (QoS) (illustrés dans le tableau A.2).

Data type properties	Domains	Ranges
ConsumabilityEfforts	RT.NFFs	String
FaultToleranceEfforts	RT.NFFs	String
MigrationabilityEfforts	RT.NFFs	String
Performance	RT.NFFs	String
ReliabilityEfforts	RT.NFFs	String
RuntimeTunning	RT.NFFs	String
ScalabilityEfforts	RT.NFFs	String
SecurityEfforts	RT.NFFs	String
StandardizedEfforts	RT.NFFs	String

TAB. A.2 : Les propriétés de la catégorie RT selon l'ontologie CloudFNF

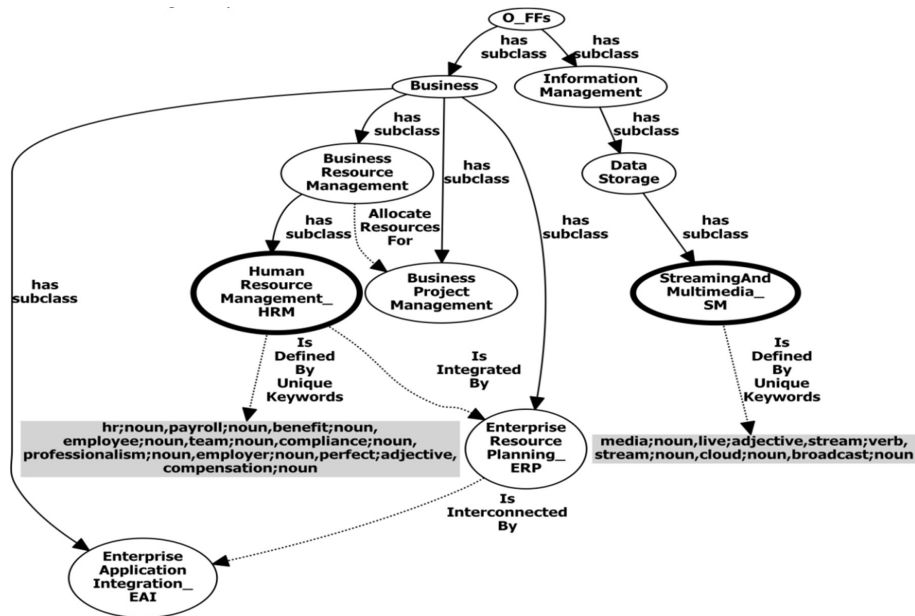


FIG. A.15 : Un extrait de l'ontologie CloudFNF se concentrant sur les FFs HRM et SM

- **Dictionnaire Cloud basé sur BabelNet :**

Il englobe des synonymes des termes anglais qui composent les descriptions textuelles des services Cloud les plus existants. Ces termes ont été vérifiés et leurs synonymes ont été collectés à l'aide de BabelNet. Les entrées de ce dictionnaire se présentent sous la forme « Token-Nom ;Type », où Type \in {nom, verbe, adjectif, expression, etc.}. En d'autres termes, "stream ;nom" et "stream ;verbe" sont deux entrées différentes. Ces synonymes seront classés selon BabelNet en concepts de domaine et en entités nommées de domaine. Comme le montre la figure (A.16), les termes et leurs synonymes seront stockés dans le dictionnaire Cloud au format JSON, où token_i représente le nom du terme et $i \in \{1: \text{nombre de tous les termes}\}$ et $cd \in \{\text{tous les concepts de l'ontologie Cloud référencée}\}$.

```

{
  token_i;Type:
  {
    "Concept": [c1,c2,...],
    "Entity": [e1,e2,...],
    "CloudDomainName": "cd"
  }
}

```

FIG. A.16 : Structure des données de Dictionnaire Cloud au format JSON

3. BDR

La figure (A.17) fournis un extrait du modèle BDR développé des NFF couverts dans l'ontologie CloudFNF en se concentrant sur les NFF, qui sont dédiés à la fois aux FF SM et HRM.

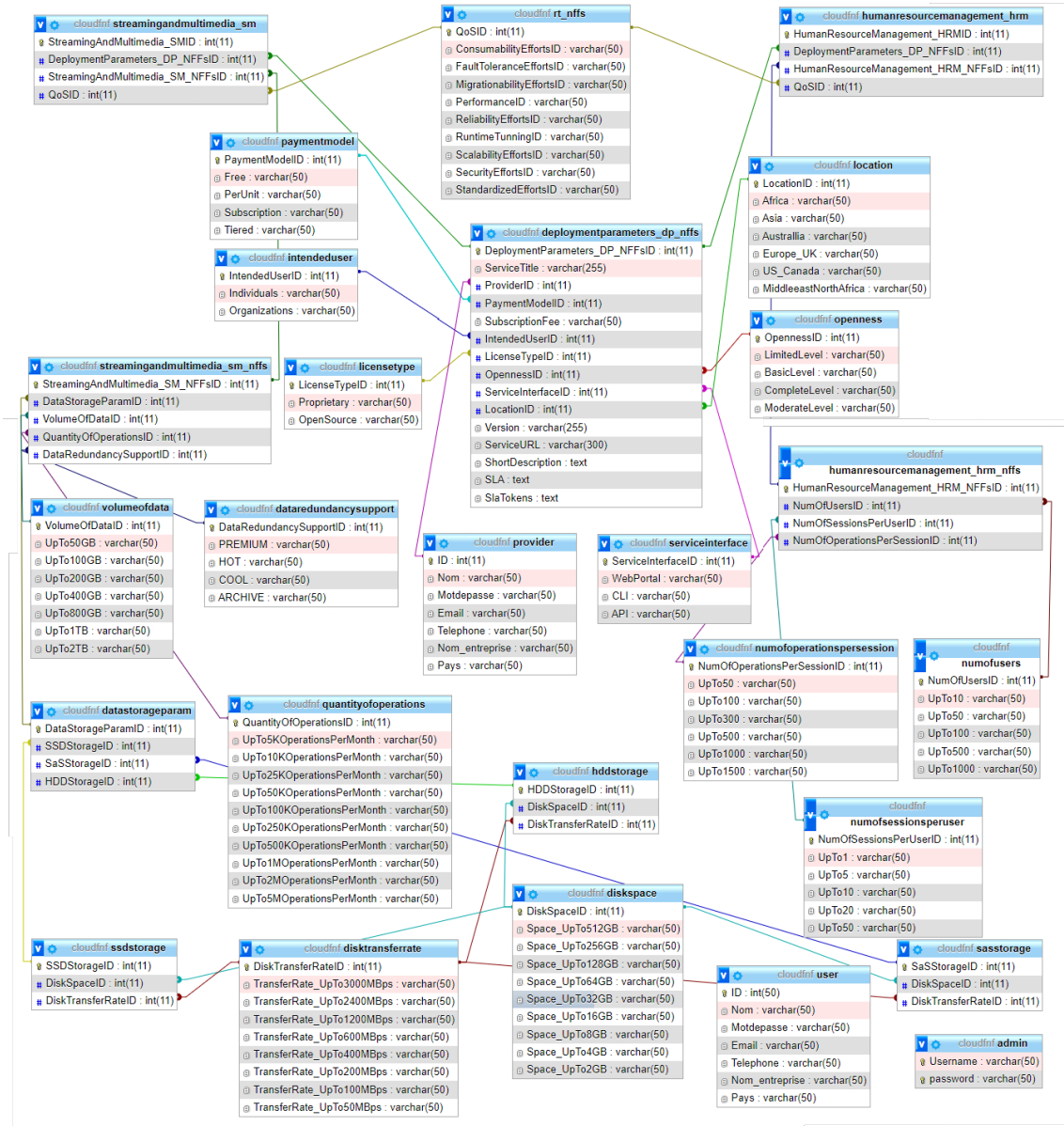


FIG. A.17 : Un extrait de BDR se concentrant sur les FFs HRM et SM

Pour construire une structure compatible pour le modèle BDR, chaque concept ontologique FF est représenté par un tableau de quatre lignes, La première ligne c'est la clé primaire nommée en ajoutant le suffixe "ID" au nom du concept ontologique FF et les trois autres lignes sont dédiées aux trois types de NFF liés au concept FF (c'est-à-dire

DeploymentParameters_DP_NFFs, RT.NFFs et le groupe exclusif des NFFs DT.NFFs liés au FF spécifié). Les trois dernières lignes sont également nommées en ajoutant le suffixe "ID" au concept nommé. Comme le montre la figure (A.17), les concepts FFs de Human Resource Management HRM et Streaming and Multimedia SM sont représentés dans le modèle BDR par deux tables nommées avec les mêmes noms de concepts FFs avec quatre lignes. Par exemple, le concept FF "Human Resource Management HRM" est représenté par le tableau "humanresourcemanagemen_hrm" qui comprend les colonnes "HumanResourceManagement_HRMID", "DeploymentParameters_DP_NFFsID", "QoSID" et "HumanResourceManagement_HRM_NFFsID".

4. Accès aux données OBDA basé sur Ontop

la couche OBDA Ontop est une couche intermédiaire entre la couche de métadonnées (en particulier le composant d'ontologie CloudFNF) et la couche BDR. Pour plus d'accessibilité et pour exploiter les avantages des modèles CloudFNF et BDR. La fonctionnalité principale de ce système est la réponse aux requêtes et le transfert des requêtes SPARQL sur l'ontologie en SQL sur BDR .

5. Découverte

La couche Découverte comprend trois composants principaux :

- **Mise à jour des mots clés et des termes de dictionnaire :**

Les mots clés uniques associés aux FFs et les termes associés à chaque domaine de concept dans le dictionnaire développé basé sur BabelNet sont très importants pour atteindre la convergence requise entre les termes anglais des utilisateurs non experts et les concepts de Cloud computing. En raison de la publicité continuée des services Cloud par les fournisseurs de Cloud dans le référentiel sémantique développé, il est très important de maintenir à jour les mots clés et le dictionnaire afin d'atteindre la précision et l'efficacité requises de cette convergence et d'augmenter l'exactitude. Les termes des descriptions textuelles (c'est-à-dire les "SLATokens") des services Cloud classés dans chaque concept FF dans la BDR sont collectés directement à partir d'OBDA et ils sont passés au processus **Mise à jour des mots clés associés**. Ce processus contient les étapes suivantes :

- **Mise à jour du dictionnaire Cloud** : Dans cette étape on ajoute les termes de SLATokens dans le dictionnaire s'ils ne sont pas déjà dans le dictionnaire, et il faut aussi que ces termes sont dans au moins 20% de services de FF.
- **Unified Termes** : Pour chaque FF, nous avons un ensemble de SLATokens, chaque SLATokens de service est représentée par une liste des termes. Ensuite, on applique le dictionnaire sur ces listes pour unifier les termes à l'aide de relations synonymes. Par exemple, tous les synonymes du terme "broadcast ;verb" dans tous les SLATokens seront remplacés par ce terme. En conséquence, tous les termes de même sens dans tous les SLATokens seront unifiés en termes d'orthographe et de sens.
- **TFWeight** : Dans cette étape on calcule TFWeight pour chaque terme, telle que TFWeight est le nombre de fois que le terme existe dans SLATokens, et on élimine les termes avec des poids mineurs.
- **FPGrowth** : L'objectif de cet algorithme est de détecter les mots-clés qui sont les termes les plus courants dans toutes les SLATokens des services pour chaque FF.

Enfin, les propriétés obtenues seront utilisées pour distinguer les services d'un concept particulier dans l'ontologie Cloud.

- **Matching** :

Le Matching est le composant central du framework proposé, où le processus de découverte commence à partir de ce composant en faisant correspondre la requête de découverte de texte des utilisateurs ou la description de service de fournisseur aux mots clés associés aux concepts FFs de l'ontologie Cloud FNF à l'aide de l'entropie relative (H_{rel}). Les FFs qui atteignent $H_{rel} \geq 60\%$ sont ajoutés à la liste "Fonctionnalités correspondantes" pour être récupérés en tant que FF les plus liés à la requête d'utilisateur ou la description de fournisseur. Pour accélérer le processus de mise en correspondance, la relation "isDefinedByMergedKeywords" est utilisée pour définir chaque concept FF en fusionnant tous les mots clés uniques associés aux sous-concepts FF du sous-réseau sémantique enraciné par ce concept FF. Comme le montre la figure (A.18), chaque FF est défini par deux types de mots clés, mots clés uniques et mots clés fusionnés. Les termes de la requête des utilisateurs ou de la description de fournisseur sont mis en correspondance avec les mots clés uniques associés au concept FF visité afin de décider

s'il est lié ou non au terme de la requête. Aussi, les mots clés fusionnés associés aux enfants de ce concept sont mis en correspondance pour décider s'il existe d'éventuels concepts apparentés ou non afin de décider d'éliminer ou non la recherche dans son sous-réseau. Le dernier type de mots-clés n'est utilisé que pour accélérer le processus de découverte, tandis que le premier type de mots-clés est utilisé pour récupérer les FFs liés au terme de requête des utilisateurs.

La sortie, est une liste de FF correspondants triée par l'entropie relative, récupérée dans l'interface utilisateur avec une courte description attachée à chaque FF pour désigner celles qui sont destinées.

Le degré de matching est mesuré à l'aide de l'entropie relative (H_{rel}) comme indiquée dans l'équation suivante :

$$H_{rel} = \sum_{i=1}^N \frac{P_i \log P_i}{\log n} \quad H_{rel} \in [0, 1] \quad (\text{A.1})$$

Où n est le nombre de propriétés du concept et P_i est la probabilité de la propriété i . En d'autres termes, H_{rel} mesure comment ses propriétés sont uniformément réparties dans la requête d'utilisateur donnée.

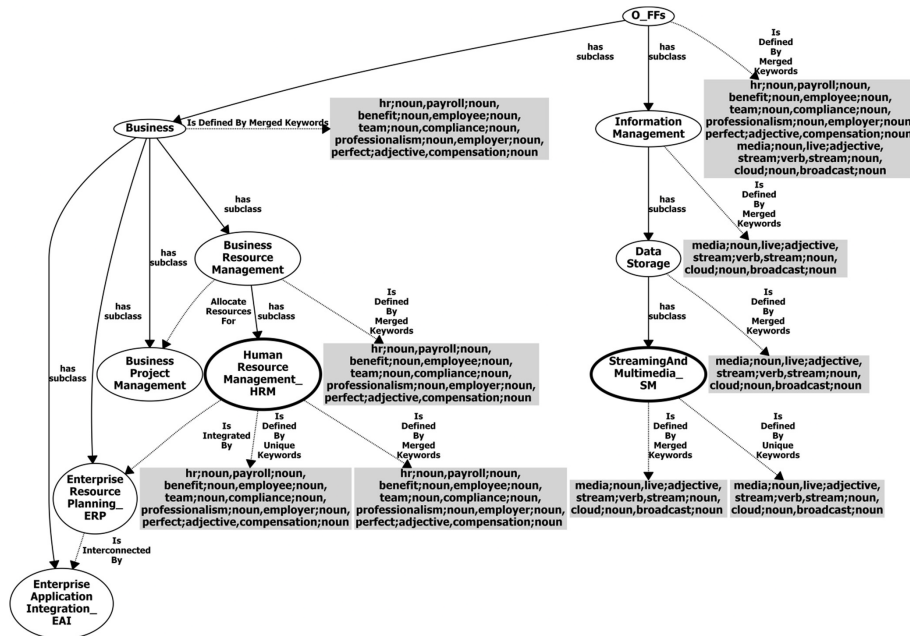


FIG. A.18 : Un extrait de l'ontologie se concentrant sur les relation Merged et UniqueKeywords sur FFs HRM et SM

- **Classement :**

En utilisant le dictionnaire Cloud basé sur BabelNet développé, les services Cloud

extraits de la couche OBDA sont classés en fonction de leur description textuelle correspondant aux degrés par rapport aux mots clés associés aux FF requis. Ces degrés de correspondance sont calculés à l'aide de l'entropie relative (H_{rel}) qui mesure la manière dont les mots clés associés de chaque FF requis sont uniformément répartis dans les descriptions textuelles des services Cloud récupérés. Ces descriptions sont également stockées sous la même forme que les entrées du dictionnaire développé (c'est-à-dire Token_Nom;Type) dans les soi-disant "SLATokens" pendant la phase de publicité des services.

le H_{rel} total pour chaque service récupéré est calculé en utilisant l'équation suivante :

$$TotalH_{rel} = - \sum_{i=1}^N \frac{P_i \log P_i}{\log n} \quad H_{rel} \in [0, 1] \quad (\text{A.2})$$

Où n est le nombre de FF requis, et P_i est le H_{rel} correspondant au FF requis. En d'autres termes, $TotalH_{rel}$ mesure comment ce service est lié à tous les FF requis.

A.3 Solution de Abbas et al [3]

Les agents utilisés dans ce SMA sont classés en deux catégories :

1. Les agents agissant pour le compte du client de Cloud :

- **Customer Agent (CA)** : Cet agent communique avec l'interface d'utilisateur pour récupérer la demande de client pour interprète la demande avant de passer à DA. trois agents peuvent communiquer avec cet agent DA, SA et RtA.
- **Discovery Agent (DA)** : Le rôle de cet agent consiste à découvrir les services existents qui correspondent à la demande de l'utilisateur du Cloud.
- **Reasoning Agent (RA)** : En cas d'échec dans la découverte de services au niveau de DA. Ce dernier passait la demande au RA pour essayer de trouver les services similaires a la demande de client.
- **Selection Agent (SA)** : Le rôle de cet agent consiste à sélectionner parmi les services récupérés lors de le découvert, le service le plus approprié à la demande de client.
- **Rating Agent (RtA)** : Cet agent modifier la valeur de réputation de service choisi par le client.

2. Les agents agissant pour le compte du fournisseur des services :

- **Provider Agent (PA)** : Le PA fournit également une interface graphique au fournisseur pour les nouveaux services et pour modifier ou supprimer les services.
- **Registry Agent (RgA)** : Le RgA validera les résultats transmis par le PA. Il contacte l'ontologie pour ajouter les services transmis par le PA ou pour la mise à jour des services.

Chaque agent a un comportement dynamique basé sur la prestation de services avec la spécification de conformité basée sur FIPA. FIPA est essentiellement une organisation standard IEEE de la société informatique. L'objectif principal de FIPA est d'établir des normes d'intercommunication entre les agents logiciels pour une interopérabilité maximisée au sein et entre les agents par un ensemble de spécifications. L'architecture de cette solution est illustrée dans la figure suivante :

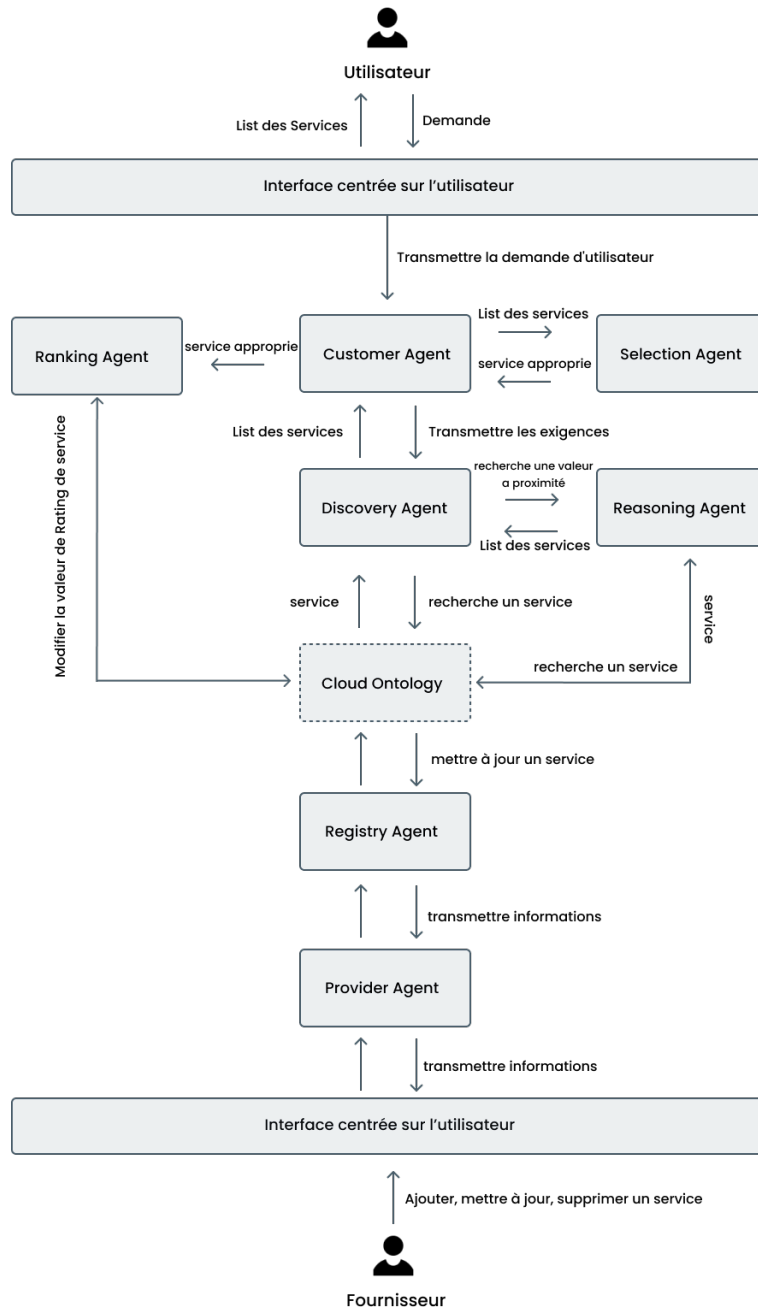


FIG. A.19 : Architecture de cadre selon la solution d'Abbas et al [2]

3. Ontologie Cloud :

L'ontologie contient des connaissances sur les entités utilisées dans le domaine du Cloud, telles que les concepts, les paramètres et les relations entre les objets dans l'environnement du Cloud. Pour comprendre les concepts, nous avons implémenté une ontologie on utilisant l'outil Protégé, Comme le montrent les figures suivantes.

(1) **Les classes** : Les concepts de la solution sont indiqués dans la figure (A.20) :

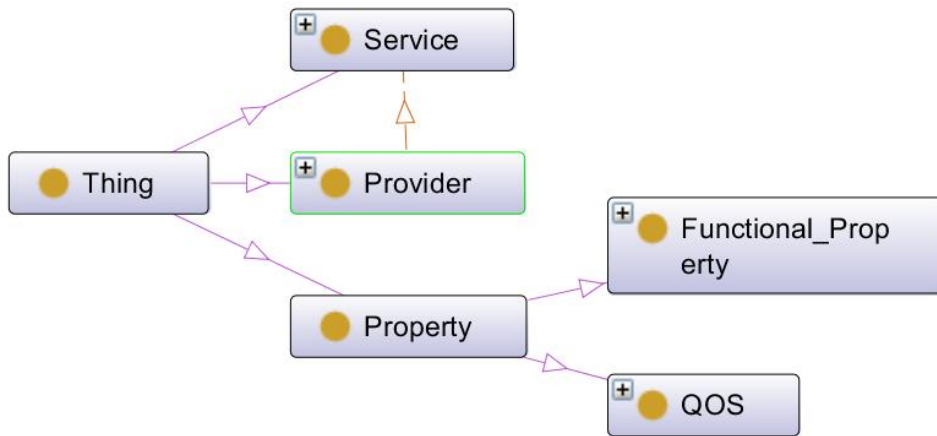


FIG. A.20 : Les différentes classes de l'ontologie

- Classe provider : C'est lui qui propose des services.
- Classe service : Chaque fournisseur propose un service qui est soit de type IaaS, PaaS ou SaaS.
- Classe Propriété : On trouve deux types de propriétés qui sont les propriétés non fonctionnelles concernant les qualités de services et les propriétés fonctionnelles.

(2) **Les relations** : Les différentes relations existantes entre les différentes classes sont présentées dans le tableau (A.3) ci-dessous :

Object properties	Domains	Ranges	Observations
Provide	Provider	Service	Symétrique, Fonctionnelle inverse
Has_FF	Service	Fonctionnal_Property	Symétrique, Fonctionnelle
Has_QOS	Service	QOS	Symétrique, Fonctionnelle

TAB. A.3 : Les différentes relations de l'ontologie

(3) **Les attributs** : Les paramètres contiennent tous les propriétés fonctionnelles et non fonctionnelles du service Cloud, comme indiqué dans le tableau (A.4) ci-dessous.

Data type properties	Domains	Ranges	Observations
IDProvider	Provider	Integer	Fonctionnelle
NameP	Provider	String	
IDService	Service	Integer	Fonctionnelle
NameS	Service	String	
URL	Service	String	
IDFF	Functionnal_Property	Integer	Fonctionnelle
BAND	Functionnal_Property	Double	
CPU	Functionnal_Property	Double	
DISK	Functionnal_Property	Double	
ECU	Functionnal_Property	Double	
OS	Functionnal_Property	Double	
RAM	Functionnal_Property	Double	
IDQOS	QOS	Integer	Fonctionnelle
Price	QOS	Double	
Rating	QOS	Double	
Availability	QOS	Double	

TAB. A.4 : Les différentes attributs de l'ontologie

Ces attributs sont utilisés par différents fournisseurs de Cloud (tels qu'Amazon, IBM, Rackspace, etc.) et sont prise en compte et mis en correspondance lors de la découverte du service Cloud par un utilisateur du Cloud.

4. Fonctionnement du SMA :

Cette section présente la méthodologie de travail complète de la solution [3], à l'aide du schéma de la figure (A.19). Pour bien comprendre les concepts abordés dans ce travail, nous avons créé une application en utilisant l'environnement JavaEE, ainsi qu'un SMA en utilisant le framework JADE pour la découverte et la sélection des services comme indiqué dans les figures ci-dessous :

- **Étape 1:** Les utilisateurs utilisent cette interface pour découvrir les services Cloud. Ils commencent leur processus de découverte en composant des requêtes de découverte par mots clés. Une fois qu'une demande d'utilisateur est reçue via une interface utilisateur contenant de nombreux paramètres fonctionnels et non fonctionnels comme montrer la figure (A.21), la demande de l'utilisateur est transmise à CA pour l'interprétation du message.

Functional Property:

Select the type of service:

Select the price range: To

Select the memory range: To

Select the Bandwidth rang: To

Select the Ecu range: To

Select the Cpu range: To

Select the Persistent Storage range: To

Select the OS:

QoS Paramaters:

Set Priority*

Price: ANY

Rating: ANY

Availability: ANY

Result

Service Provider Name	URL	Service Name	Rating	Avail	PRICE	RAM	CPU	OS	DISK	BAND	Action
-----------------------	-----	--------------	--------	-------	-------	-----	-----	----	------	------	--------

FIG. A.21 : Interface utilisateur pour la découverte des services

- Étape 2:** À partir du moment où les paramètres sont remplis par l'utilisateur. Le CA les transmet à DA. Le DA consulte l'ontologie pour trouver le service adapté à la requête de l'utilisateur Cloud. La découverte est basée sur la correspondance des services avec les paramètres entrés par l'utilisateur du Cloud service qui sont les exigences fonctionnelles de l'utilisateur telles que le stockage, la taille de la mémoire et le coût, etc.). Une fois que la découverte réussie des services Cloud appropriés le DA envoie les informations à la CA pour la sélection par la SA (comme illustrés à la figure A.22).
- Étape 3:** Une fois que le DA a terminé le processus de recherche, il renvoie la liste des services correspondant au CA. Le CA transmet les services correspondant à SA pour l'étape de sélection.

Select the price range: To

Select the memory range: To

Select the Bandwith rang: To

Select the Ecu range: To

Select the Cpu range: To

Select the Persistant Storage range: To

Select the OS:

Set Priority*

Price:

Rating:

Availability:

Result

Service Provider Name	URL	Service Name	Rating	Avail	PRICE	RAM	CPU	OS	DISK	BAND	Action
Rackspace	http://www.rackspace.com/cloud/public/pricing	general-2	3.0	90.0	0.25	2.0	2.0	Linux	10.0	100.0	Order
AmazonE2	http://aws.amazon.com/ec2/pricing	m3.medium	3.0	99.0	0.26	3.75	1.0	Linux	20.0	15.0	Order
IBM	ibm.com	test2	1.0	80.0	0.5	8.0	8.0	Linux	20.0	160.0	Order

FIG. A.22 : Résultat de découverte selon la solution de Abbas et al

- **Étape 4:** Dans le cas où le DA échoue à trouver des services, il envoie une demande au RA en consultant l'ontologie. Dans cette étape nous avons proposé notre propre raisonnement qui est basé sur la notion de probabilité. Comme indiqué dans l'algorithme (1), ce raisonnement donne des services avec une probabilité $\geq 50\%$ de correspondance avec la demande du client, il renvoie les services obtenus au DA.
- **Étape 5:** Dans cette étape l'agent SA va sélectionner parmi les services envoyés par CA le service le plus approprié aux critères QoS choisis par le client, nous avons choisi et implémenté la sélection par MCDM, donc le client saisit les valeurs des critères avec le poids de chaque critère, après l'agent SA choisit le service, ce service sera transmis à CA pour l'affichage du service dans l'interface comme le montre la figure (A.23).
- **Étape 6:** Le client peut donner son avis à propos du service qu'il utilise à travers l'agent RtA, cet agent travaille pour modifier la valeur de réputation du service après chaque saisie dans l'évaluation du service.

Functional Property:

Select the type of service:

Select the price range: To

Select the memory range: To

Select the Bandwidth rang: To

Select the Ecu range: To

Select the Cpu range: To

Select the Persistant Storage range: To

Select the OS:

QoS Paramaters:

Set Priority*

Price:

Rating:

Availability:

Result

Service Provider Name	URL	Service Name	Rating	Avail	PRICE	RAM	CPU	OS	DISK	BAND	Action
Proximus	http://www.proximus.com/telecom/telecom	proximus	3.0	100	0.05	20	20	linux	100	1000	Order

FIG. A.23 : Interface pour la sélection des services Cloud

- **Étape 7:** Le PA interagit avec l'interface graphique conviviale qui comprend de nombreux attributs fonctionnels et non fonctionnels. Le fournisseur de services doit mettre à jour les paramètres QoS. Le PA fournit une interface graphique pour des nouveaux services et pour modifier ou supprimer les services.
- **Étape 8:** Le RgA valide les informations transmises par PA pour stocker, modifier ou supprimer les services dans l'ontologie

Le Java Agent Development Framework (JADE) est un cadre logiciel utilisé pour l'application de l'agent et la conformité aux spécifications FIPA pour les SMA interopérables. Dans cette solution, le JADE est utilisé pour démontrer le cadre SMA proposé pour la découverte et la sélection du service Cloud. Le JADE comprend un ensemble de conteneurs qui fournissent des services de niveau système pour le fonctionnement du SMA. Chaque conteneur d'agent dispose d'un registre RMI (Remote Method Invocation) qui gère et contrôle le cycle de vie des agents en les créant, les supprimant, les suspendant et les reprenant.

Le SMA pour chaque agent est construit comme présenté dans la Figure (A.24).

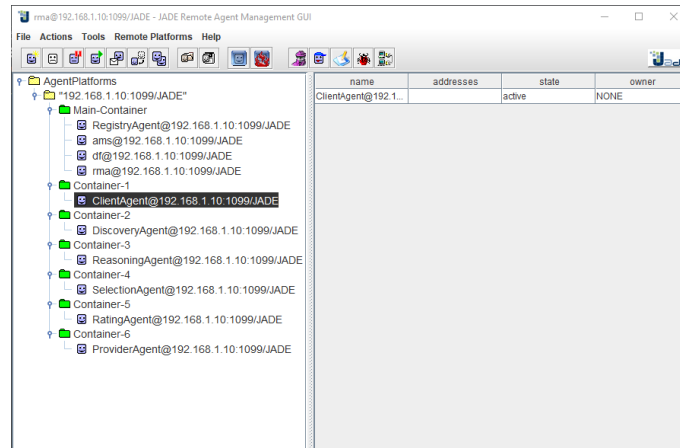


FIG. A.24 : L'interface JADE selon la solution de Abbas et al

Il y a sept conteneurs et un conteneur principal composé de RgA, AMS, DF et RMA. L'application JADE fournit un service complet aux agents tels que la transmission de messages, la migration vers les agents et la gestion des ressources des agents. La nature de la tâche de chaque agent dépend du comportement de l'agent qui doit être défini dans le système JADE. Après l'exécution de la solution, la figure (A.25) montre l'échange des messages ACL entre les agents afin d'obtenir le résultat.

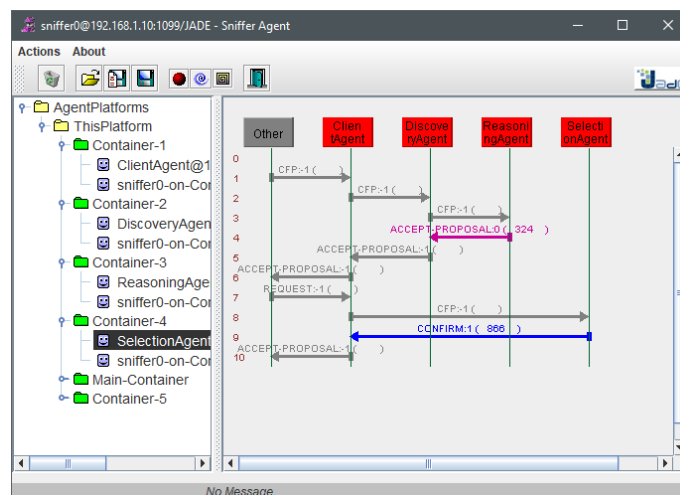


FIG. A.25 : Interface JADE montre la communication entre les agents de SMA

A.4 Solution de Kang et al [45]

Cette solution présente un protocole de découverte de services Cloud basé sur un SMA. Ce système Multi-agent est composé de 3 agents :

- **User (utilisateur) Agent** : Un agent utilisateur fournit une interface pour l'utilisa-

teur. Il affiche les demandes aux agents Broker à l'aide de la transmission de messages. Si la procédure de connexion est terminée, il affiche les résultats à un utilisateur via une interface utilisateur.

- **Provider (fournisseur) Agent** : Les agents fournisseurs ont des fonctionnalités similaires aux agents utilisateurs, mais agissent pour le compte de fournisseurs humains.
- **Broker (middleware) Agent** : Un agent Broker relie les agents utilisateurs et fournisseurs en utilisant un algorithme de connexion. Un agent Broker peut envoyer un message de recommandation basé sur les données historiques d'autres agents Brokers dans la base de données, de sorte que l'agent Broker peut recommander d'autres agents Brokers aux agents utilisateurs qui n'ont pas réussi à se connecter aux agents fournisseurs.

L'architecture de cette solution est illustrée à la figure (A.26).

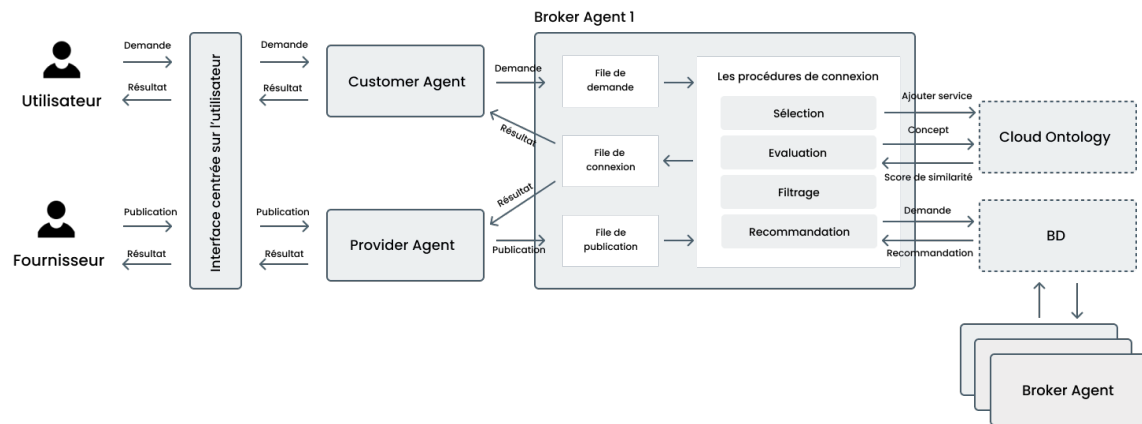


FIG. A.26 : Architecture de cadre selon la solution de Kang et al [45]

1. Broker système pour les services Cloud :

Le Broker système pour les services Cloud comprend plusieurs Broker agents et agents commerciaux (agents utilisateurs et agents fournisseurs). La procédure de courtage de services Cloud ainsi le rôle de chaque agent utilisé dans le système de courtage est illustrée à la figure (A.27).

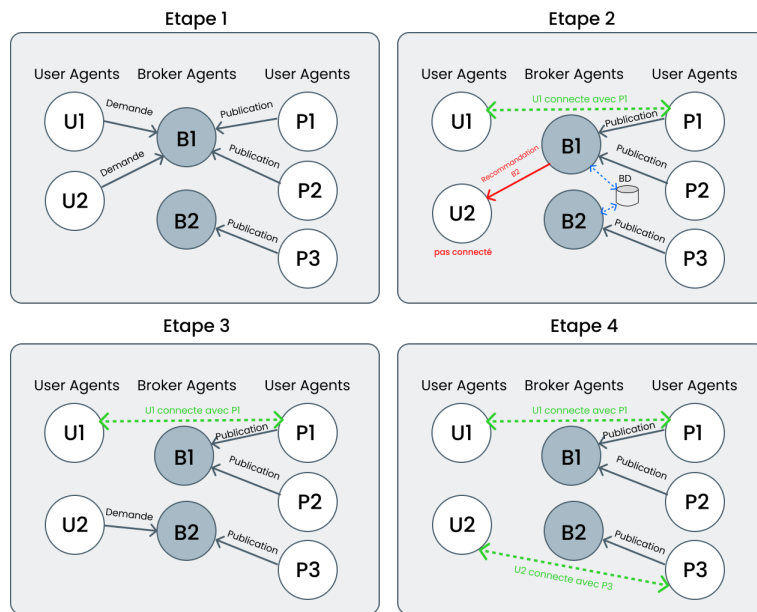


FIG. A.27 : La procédure de courtage de services cloud basé sur des agents

Le Broker système pour les services de Cloud fonctionne comme suit :

Dans un premier temps, les agents utilisateurs (respectivement les agents fournisseurs) envoient les requêtes (respectivement les annonces) à un agent Broker. La figure (A.28) montre la requête et l'annonce envoyée par l'utilisateur respectivement fournisseur à l'agent Broker.

<p>Functional Property For User:</p> <p>Select the type of service: <input type="text" value="IaaS"/></p> <p>Select the price range: <input type="text" value="4500"/></p> <p>Select the memory range: <input type="text" value="500"/></p> <p>Select the Storage range: <input type="text" value="60"/></p> <p>Select the Cache rang: <input type="text" value="10"/></p> <p>Select the Bandwidth rang: <input type="text" value="0.5"/></p> <p>Select the Latency range: <input type="text" value="50"/></p> <p>Select the Time range: <input type="text" value="20-30"/></p> <p>Select the OS: <input type="text" value="Windows"/></p> <p>Select the CPU: <input type="text" value="Intel"/></p> <p><input type="button" value="Discover"/></p> <p style="text-align: center;">Result</p>	<p>Functional Property For Provider:</p> <p>Select the type of service: <input type="text" value="CaaS"/></p> <p>Select the price range: <input type="text" value="5000"/></p> <p>Select the memory range: <input type="text" value="600"/></p> <p>Select the Storage range: <input type="text" value="70"/></p> <p>Select the Cache rang: <input type="text" value="20"/></p> <p>Select the Bandwidth rang: <input type="text" value="0.8"/></p> <p>Select the Latency range: <input type="text" value="60"/></p> <p>Select the Time range: <input type="text" value="20-30"/></p> <p>Select the OS: <input type="text" value="Windows"/></p> <p>Select the CPU: <input type="text" value="Intel"/></p> <p><input type="button" value="Discover"/></p> <p style="text-align: center;">Result</p>
---	---

FIG. A.28 : Interface utilisateur et fournisseur

Deuxièmement, l'agent Broker vérifie si les files d'attente d'annonces et de demandes sont vides. Dans le cas où ces files d'attente ne sont pas vides, l'agent Broker exécute des procédures de connexion qui consistent principalement en quatre étapes (sélection, évaluation,

filtrage et recommandation). Dans le cas où ces files d'attente sont surchargées, l'agent Broker envoie ces demandes (respectivement, les annonces) à d'autres agents Broker pour équilibrer les charges de travail.

Troisièmement, après avoir exécuté la procédure de connexion, l'agent Broker envoie le résultat aux agents utilisateur et fournisseur.

Quatrièmement, si un agent utilisateur ne parvient pas à se connecter à un agent fournisseur, l'agent Broker recommande un autre agent Broker avec le plus grand potentiel dans le courtage via la base de données afin que l'agent utilisateur puisse envoyer une demande à un autre agent Broker.

Cinquièmement, si la demande d'un agent utilisateur correspond à la publicité d'un agent fournisseur géré par un autre agent Broker, la connexion entre l'agent utilisateur et l'agent fournisseur sera établi avec succès. Sinon, l'agent Broker recommande d'autres agents Broker potentiels à l'agent utilisateur.

S'il n'y a pas d'agent Broker permettant à un agent utilisateur d'établir une connexion avec des agents du fournisseur dans la base de données, la connexion échoue.

2. Ontologie Cloud :

L'ontologie Cloud qui est utilisée dans cette solution est constituée de concepts. Le concept "CloudSystem" a cinq nœuds enfants différents :

- Le IaaS fournit des ressources de calcul (par exemple, l'EC2 d'Amazon fournit des machines virtuelles).
- Le PaaS fournit aux développeurs Cloud des environnements de niveau programmation (par exemple, Google App Engine fournit un environnement d'exécution python et une API permettant aux applications d'interagir avec l'environnement d'exécution Cloud de Google).
- Le SaaS fournit un logiciel à usage spécifique, accessible à distance via Internet (par exemple, le CRM de Salesforce).
- Le DaaS permet aux utilisateurs de stocker des données sur des disques distants (par exemple, le S3 d'Amazon).
- Le CaaS fournit une bande passante dédiée, la sécurité du réseau, le chiffrement et la surveillance du réseau.

Avec cette ontologie, la similitude entre les services Cloud peut être déterminée.

Les deux tableaux suivants (A.5), (A.6) représentent respectivement les différents attributs ou les propriétés de donnée (Data Properties) et les relations (Object Properties) des différents concepts de l'ontologie :

Data type properties	Domains	Ranges
hasID	CloudSystem	Integer
hasCache	CloudSystem	Double
hasMemory	CloudSystem	Double
hasNetworkBandWidth	CloudSystem	Double
hasNetworkLatency	CloudSystem	Double
hasStorage	CloudSystem	Double
Price	CloudSystem	Double
TimeSlot	CloudSystem	String
hasCPUValue	CPU	String
hasOSValue	OS	String

TAB. A.5 : Les propriétés de l'ontologie

Object properties	Domains	Ranges
hasCPU	CloudSystem	CPU
hasOS	CloudSystem	OS

TAB. A.6 : Les relations de l'ontologie

3. Connecter les utilisateurs et les fournisseurs :

La procédure de connexion (sélection, évaluation, filtrage et recommandation) entre les utilisateurs et les fournisseurs est présentée comme le montre la figure (A.26).

- **Phase de sélection :**

Lors de la phase de sélection, la demande des exigences minimales des agents utilisateurs et les annonces des exigences maximales des agents fournisseurs seront comparées par l'agent Broker. Cette étape garantit que le fournisseur de services sélectionné répond aux exigences minimales demandées par l'utilisateur. Par exemple :

$$Critres = (Has_Storage_u \leq Has_Storage_p) \wedge (Has_Memory_u \leq Has_Memory_p).$$

Où $Has_Storage_u$ est la valeur de stockage minimale demandé par l'utilisateur et $Has_Storage_p$ est la valeur maximale de stockage publiée par le fournisseur. Si cette comparaison n'est pas vérifiée la connexion entre l'utilisateur et le fournisseur ne sera pas sélectionnée, sinon elle sera sélectionnée. De cette manière, les utilisateurs sont plus susceptibles de choisir des fournisseurs appropriés dont les ressources disposent de Stockage et Mémoire inférieur ou égale à l'exigence maximale de fournisseurs pour pouvoir exécuter l'application. Par conséquent, l'objectif est de choisir le fournisseur le plus approprié au lieu de sélectionner le fournisseur qui a la capacité la plus élevée.

- **Phase d'évaluation :**

Après la fin de la phase de sélection, nous avons un ensemble de connexions potentielles des agents utilisateurs et fournisseurs. Au cours de l'étape d'évaluation, chaque connexion est évaluée en fonction de : raisonnement de similarité, utilitaires de prix et de plage horaire correspondants.

- 1. **Raisonnement de similarité :**

La similarité entre la demande des utilisateurs et les annonces des fournisseurs peut être déterminée par :

- Raisonnement de similarité de concept
- Raisonnement de similarité de propriété d'objet
- Raisonnement de similarité de propriété de type de données.

Comme suit :

$$Sim(p, q) = \alpha Sim_{con}(p, q) + \beta Sim_{obj}(p, q) + (1 - \alpha - \beta) Sim_{data}(p, q)$$

α, β sont les poids de chaque clause, p représente l'individu de l'utilisateur, q représente l'individu du fournisseur, et la plage de la valeur évaluée est $0 \leq Sim(p, q) \leq 1$.

- 1.1 **Raisonnement de similarité de concept :**

La similarité de concept peut être déterminée comme suit :

$$Sim_{con}(p, q) = \frac{|Super(C^p) \cap Super(C^q)|}{|Super(C^p)|}$$

C^p et C^q sont les concepts les plus spécifiques auxquels appartiennent respectivement les individus p et q, et $Super(C^p)$ (respectivement, $Super(C^q)$) est un ensemble de tous les super-concepts accessibles à partir du concept C^p (respectivement, le concept C^q). Par exemple, $Super(PaaS)$ est l'ensemble des concepts 'PaaS', 'CloudSystem', 'Thing'

qui sont accessibles à partir du concept 'PaaS' comme illustré dans la Figure (??).

1.2 Raisonnement de similarité de propriété d'objet :

La similarité des propriétés d'un objet peut être déterminée comme suit :

$$Sim_{obj}(p, q) = \frac{\sum_{(x,y) \in U} Sim(x, y)}{|O(p)|},$$

$$U = \{(x, y) | (p, r, x) \in O(p), (q, r, y) \in O(q)\}$$

$O(p)$ est un ensemble de triplets contenant les propriétés d'objet de l'individu p , et p est le sujet. Chaque triplé est composé de : sujet, prédicat, une valeur d'objet pour exprimer l'ontologie.

U est l'ensemble des valeurs d'objet qui a le prédicat commun r des individus p et q dans chaque triplet $O(p)$ et $O(q)$, respectivement.

1.3 Raisonnement de similarité de propriété de type de données :

La similarité des propriétés de type de données peut être déterminée comme suit :

$$Sim_{data}(p, q) = \frac{\sum_{(x,y,r) \in V} Comp(x, y, r)}{|D(p)|},$$

$$V = \{(x, y, r) | (p, r, x) \in D(p), (q, r, y) \in D(q)\},$$

$$Comp(x, y, r) = 1 - \frac{|(x - y)|}{MAX_{distance}(x, r)},$$

$$MAX_{distance}(x, r) = max_{i \in I(r)} (|x - i|),$$

$$I(r) = \{i | (s, r, i) \in Ontologie\}$$

où $D(p)$ est un ensemble de triplets qui contient les propriétés de type de données de l'individu p et q . Chaque triplet est composé de : sujet, prédicat, une valeur de type de données pour exprimer l'ontologie.

1.4 Exemple de raisonnement par similarité :

Grâce aux trois méthodes de similarité ci-dessus, nous pouvons déterminer la similitude entre deux individus. Pour bien expliquer le raisonnement par similarité nous allons utiliser un exemple illustratif :

Supposons que Service1 et Service2 soient des individus dans les concepts IaaS et CaaS, respectivement, dans une ontologie Cloud représentant. En outre, nous supposons que Service1 et Service2 ont certaines propriétés. Le tableau (A.7) montre les concepts et leurs individus, et le tableau (A.8) montre ces individus et leurs propriétés.

Concept	Individu
Iaas	Service1
Caas	Service2
CPU	CPU1
CPU	CPU2
OS	OS1
OS	OS2

TAB. A.7 : Exemple de concept et leurs individus

Individu	Data type properties	Valeur
Service1	hasNetworkLatency	10.0
Service1	Price	5000.0
Service1	hasNetworkBandwith	0.5
Service1	hasCPU	CPU1
Service1	hasOS	OS1
Service1	hasMemory	600.0
Service1	Timeslot	10-11
Service1	hasStorage	80.0
Service1	hasCache	20.0
Service2	hasNetworkLatency	9.0
Service2	Price	4500.0
Service2	hasNetworkBandwith	0.2
Service2	hasCPU	CPU2
Service2	hasOS	OS2
Service2	hasMemory	500.0
Service2	Timeslot	10-11
Service2	hasStorage	60.0
Service2	hasCache	10.0
CPU1	hasCPUValue	Intel
CPU2	hasCPUValue	Intel
OS1	hasOSValue	Windows
OS2	hasOSValue	Windows

TAB. A.8 : Exemple d'individus leur Data type et leur valeur

On commence par raisonnement de similarité de concept :

Pour calculer $Sim_{con}(Service1, Service2)$, nous savons que :

- $|Super(IaaS)| = 3$
- $|Super(CaaS)| = 3$
- $|Super(IaaS) \cap Super(CaaS)| = 3$

Par conséquent, la similitude de concept est : $Sim_{con}(Service1, Service2) = 3/3 = 1$

Raisonnement de propriété d'objet :

Pour calculer $Sim_{obj}(Service1, Service2)$, nous savons que :

- $|O(p)| = 2$
- L'ensemble des valeurs des propriétés d'objet communes de Service1 et Service2 est $U = (CPU1, CPU2), (OS1, OS2)$

La similitude de chacun des membres de U peut être déterminée par une procédure récursive. En utilisant les trois types de méthodes de raisonnement de similarité qui ont été expliquées précédemment et seront expliquées, nous pouvons donner les valeurs de similarité de chaque membre de U, qui sont $Sim(CPU1, CPU2), Sim(OS1, OS2)$. Puisque CPU1 et CPU2 n'ont pas leurs types de propriété d'objet (voir Tableau (A.8)), seuls $Sim_{con}(CPU1, CPU2)$ et $Sim_{data}(CPU1, CPU2)$ sont calculés. Pareil pour OS1 et OS2. Sachant que la propriété de type de donnée pour CPU et OS n'est pas numérique, donc on fait une comparaison exacte entre les propriétés. En conséquence, nous pouvons obtenir les similitudes de chaque membre, qui sont :

$$Sim_{obj}(Service1, Service2) = 1$$

Raisonnement de similarité de propriété de type de données :

Pour la similarité des propriétés de type de données, nous savons que :

- $|D(Service1)| = 5$
- L'ensemble des valeurs de la propriété de type de données commune entre deux individus, qui est $V = \{(600, 500, hasMemory), (20, 10, hasCache), (80, 60, hasStorage), (10, 9, hasNetworkLtcency), (0.5, 0.2, hasNetworkBandwith)\}$.

La similarité numérique pour le premier membre de l'ensemble V est la suivante :

$$Comp(600, 500, hasMemory) = 1 - \frac{|(600 - 500)|}{\max(|600 - 500|, |600 - 8000|)} = 0.98$$

où la plage de mémoire est de 500 à 8000 (voir Tableau (A.9)). La similarité numérique pour les autres membres peut être calculée de la même manière, qui sont

Comp(20, 10, hasCache)=0.99, Comp(80, 60, hasStorage)=0.97, Comp(10, 9, hasNetworkLatency)=0.99, Comp(0.5, 0.2, hasNetworkBandwidth)=0.25. Par conséquent, la similarité des propriétés de type de données peut être calculée comme suit :

$$Sim_{data}(Service1, Service2) = (0.98 + 0.99 + 0.97 + 0.99 + 0.25)/5 = 0.83$$

Nous supposons que α et β sont 1/3 chacun, le même poids. Enfin, la similitude entre Service1 et Service2 peuvent être calculés comme suit :

$$Sim(Service1, Service2) = 1 * 1/3 + 1 * 1/3 + (1 - 1/3 - 1/3) * 0.83 = 0.93$$

Propriété	Intervalle
hasNetworkBandwidth	[0.1-10]
hasNetworkLatency	[1-5000]
hasCache	[1-5000]
Price	[4500-10000]
hasMemory	[500-8000]
hasStorage	[50-1000]

TAB. A.9 : Intervalle de chaque propriété

2. Utilitaires de prix et de plage horaire correspondants :

Le prix et l'utilité de créneau horaire de chacune des connexions entre un utilisateur et un fournisseur est déterminé sur la base de la correspondance des prix et de l'utilité de créneaux horaires. Avec les méthodes de correspondance suivantes, un utilisateur peut facilement découvrir les services Cloud. L'utilisateur peut également utiliser les services Cloud à un moment plus approprié.

2.1 Matching des prix des utilitaires :

L'utilité du prix $U(P)$ peut être déterminée comme suit :

$$U(P) = \frac{P_{max} - P_{min}}{P_{max}}$$

$U(P)$ peut être déterminé par le prix maximum acceptable, P_{max} , de l'utilisateur et le prix minimum acceptable, P_{min} , du fournisseur. Si P_{min} est proche de P_{max} , $U(P)$

sera faible car l'utilisateur est plus susceptible d'avoir une plus petite différence entre P_{max} (prix initial) et le prix convenu. Si P_{min} est éloigné de P_{max} , $U(P)$ sera élevé car l'utilisateur est plus susceptible d'avoir une grande différence entre P_{max} initial et le prix convenu.

2.2 Matching des utilitaires de plage horaire :

L'utilitaire de créneau horaire $U(TS)$ peut être déterminé comme suit :

$$U(TS) = \frac{TS_{user} \cap TS_{provider}}{TS_{user}}$$

$U(TS)$ peut être déterminé par la plage de créneaux horaires d'un utilisateur et la plage de créneaux horaires d'un fournisseur. Si le créneau horaire commun entre l'utilisateur et le fournisseur est proche du créneau horaire de l'utilisateur, alors $U(TS)$ serait élevé. Sinon, $U(TS)$ serait faible.

- **Phase de filtrage :**

L'étape précédente attribue la valeur de la mesure de l'utilité pour chaque connexion. Cette étape filtre les connexions entre les agents utilisateurs et les agents fournisseurs qui donnent des utilités inférieures au seuil β spécifié par l'utilisateur. La valeur de β détermine la satisfaction de l'utilisateur vis-à-vis de la publicité. Alors qu'une petite valeur de β peut conduire à un temps de connexion rapide et à davantage de fournisseurs mais avec des utilités moindres, une grande valeur de β entraîne une diminution du nombre de fournisseurs mais avec des utilités plus élevées.

- **Phase de la recommandation :**

Après l'étape de filtrage, certains des agents utilisateurs peuvent ne pas être mis en correspondance avec la ou les ressources appropriées. A ce stade, une stratégie heuristique est appliquée pour trouver un autre agent Broker qui a le plus grand potentiel pour connecter les utilisateurs aux fournisseurs les plus appropriés. Il existe deux méthodes pour déterminer l'agent Broker qui a le potentiel le plus élevé qui sont : R1 (ancienne approche de recommandation), R2 (nouvelle approche de recommandation).

1. **Ancienne approche de recommandation (R1) :**

Dans cette étape, l'agent Broker X dresse une liste des requêtes des agents utilisateurs qui s'y connectent et qui n'ont pas pu être mises en correspondance après l'étape de

filtrage. L'agent Broker X accède à la base de données pour obtenir une suggestion d'agent Broker potentiel pour chaque demande. Avec chaque demande, le Broker X examine la valeur d'attribut prédite des publicités pour recommander un autre agent Broker qui a un fournisseur avec la valeur de similarité la plus élevée. Les informations sur le nouveau Broker seront renvoyées à l'agent utilisateur par le Broker X. Après avoir trouvé le nouvel agent Broker qui a le potentiel le plus élevé, l'agent utilisateur va envoyer une requête à ce nouvel agent Broker et démarrer un nouveau cycle.

Exemple de l'ancienne approche de recommandation (R1) :

Liste des utilisateurs en échec = { U4, U8, U20.....}

Similarité la plus élevée

Provider Agent	Sim(U4,P(t))	Broker Agent
P8	0.94	B3
P12	0.91	B7
P30	0.88	B2

FIG. A.29 : Exemple de ancienne approche de recommandation (R1)

2. Nouvelle approche de recommandation (R2) :

Le problème potentiel de l'ancienne approche de recommandation est que la similarité la plus élevée entre l'utilisateur et le fournisseur avec des valeurs d'attribut prédites ne garantit pas que l'utilisateur réussit à trouver un fournisseur satisfaisant car la similarité est déterminée sur la base de la valeur prédite. Au lieu de recommander un autre agent Broker qui a le fournisseur potentiel avec la plus grande similarité basée sur les valeurs d'attributs prédites, l'agent Broker recommande l'utilisateur à un autre agent de Broker potentiel qui a la similarité moyenne la plus élevée de tous les agents de fournisseur avec des valeurs d'attribut prédites qui s'y connectent. Par conséquent, dans cette nouvelle approche de recommandation, l'agent utilisateur est plus susceptible de trouver d'autres agents fournisseurs satisfaisants dans l'agent Broker potentiel sans autre recommandation au lieu de trouver un seul agent fournisseur potentiel qui peut ne pas être satisfaisant dans l'agent Broker.

Exemple de la nouvelle approche de recommandation (R2) :

Liste des utilisateurs en échec = { U4, U8, U20.....}

Similarité la plus élevée

Broker Agent (B)	Provider Agents dans B	Moyenne Sim(U4, P(1..k) in B
B3	P11, P9, P21,	0.88
B7	0.86
B2	0.85

FIG. A.30 : Exemple de nouvelle approche de recommandation (R2)

A.5 Solution de Kwang [46]

L'architecture de cette solution est illustrée dans la figure (A.31)

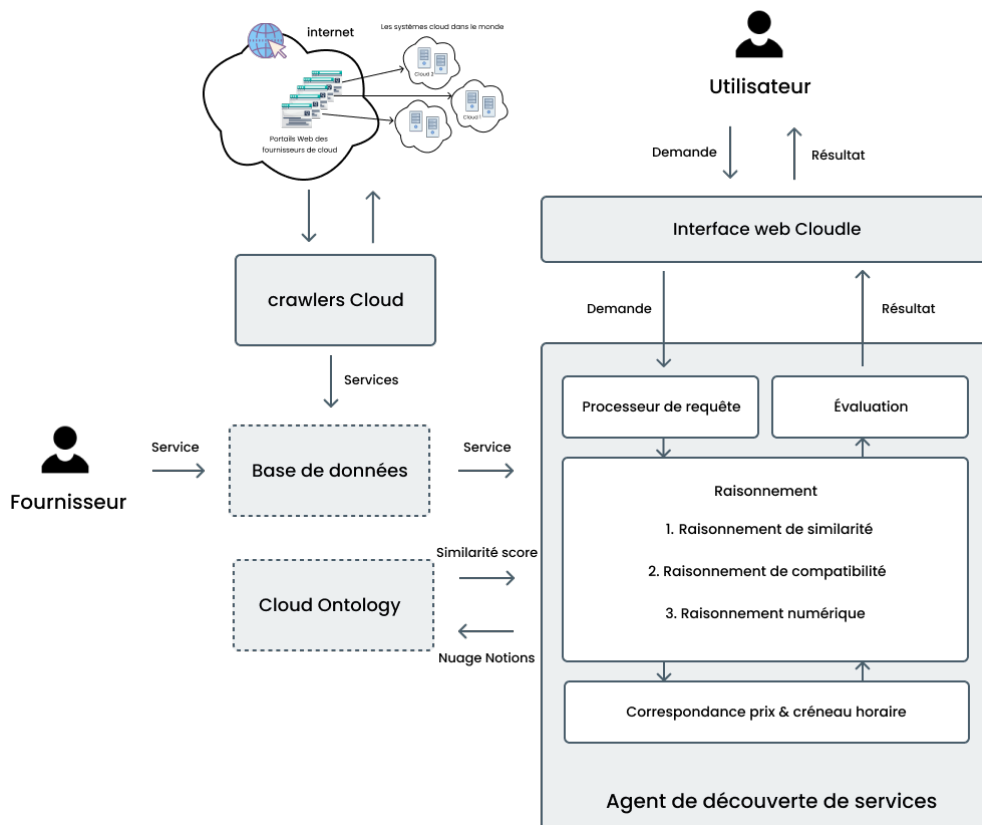


FIG. A.31 : Architecture de cadre selon la solution de Kwang [46]

1. Interface Web Cloudle :

Les consommateurs utilisent l'interface Web de Cloudle pour saisir leurs demandes de services Cloud. L'entrée Cloudle comprend les exigences fonctionnelles, techniques et budgétaires des

consommateurs pour les services Cloud. L'interface affiche aussi une liste des services évaluée comme résultat de découverte pour le consommateur.

2. Agent de découverte de services :

Le SDA comporte quatre sous-modules :

1. Un processeur de requêtes :

À l'aide de ce processeur de requêtes, le SDA extrait les mots clés essentiels tels que les concepts Cloud et les spécifications de prix et de calendrier des requêtes des consommateurs.

2. Un module de raisonnement de service :

Le SDA consulte l'ontologie Cloud pour raisonner sur la similitude entre les exigences fonctionnelles et techniques d'un consommateur et les spécifications fonctionnelles et techniques des services des fournisseurs enregistré dans la BD. En utilisant l'ontologie Cloud, le SDA effectue : un raisonnement de similarité, un raisonnement de compatibilité, et un raisonnement numérique.

2.1. Raisonnement de similarité :

Le raisonnement de similarité consiste à calculer la similarité entre deux concepts en calculant les super noeuds communs accessibles. Il peut être déterminée comme suit :

$$Sim(p, q) = \frac{|Super(C^p) \cap Super(C^q)|}{|Super(C^p)|} + (1 - \alpha) \cdot \frac{|Super(C^p) \cap Super(C^q)|}{|Super(C^q)|} \quad (A.3)$$

Où $\alpha \in [0, 1]$ détermine le degré d'influence des généralisations. C^p et C^q sont les concepts les plus spécifiques auxquels appartiennent respectivement les individus p et q , et $Super(C^p)$ (respectivement, $Super(C^q)$) est un ensemble de tous les super-concepts accessibles à partir du concept C^p (respectivement, le concept C^q).

2.2. Raisonnement de compatibilité :

Le raisonnement de compatibilité est conçu pour comparer deux nœuds frères dans une ontologie Cloud, par exemple, déterminer la compatibilité entre deux versions différentes d'un logiciel. La raison d'être de l'élaboration d'un raisonnement de

compatibilité est qu'en utilisant le raisonnement de similarité, le SDA ne peut pas raisonner sur les différences entre les concepts frères. Par exemple, dans l'ontologie partielle de Windows de la Figure (A.32) :

$$Super(Windows98) = Super(Windows7) = Super(WindowsVista) = 2.$$

$$Super(Windows98) \cap Super(Windows7) \cap Super(WindowsVista) = 1.$$

$$Sim(Windows98, WindowsVista) = 1/2$$

$$Sim(Windows98, Windows7) = 1/2$$

$$Sim(WindowsVista, Windows7) = 1/2$$

Mais les concepts frères ne différeront que par leur ordre chronologique, la fonction de mesure de la compatibilité de deux concepts p et q consiste à : 1) mesurer le degré de similarité entre p et q, et 2) différenciation entre p et q en fonction de leur ordre chronologique comme suit :

$$Compat(p, q) = Sim(p, q) + \frac{0.8^{|c_p - c_q|}}{10}$$

c_p and c_q represent the chronological orderings of different versions of a software. Par exemple, comparé à WindowsVista, Windows95 aura moins de fonctionnalités compatibles avec Windows7. Donc Le composant le plus essentiel dans l'équation est le terme $|c_p - c_q|$. Quand $|c_p - c_q|$ est grand (respectivement petit), p et q sont moins (respectivement plus) compatibles.



FIG. A.32 : Une ontologie Windows partielle

2.3. Raisonnement numérique :

Dans le raisonnement numérique, le SDA raisonne sur la similitude entre deux concepts numériques (par exemple, la vitesse du processeur et la taille de la mémoire) par rapport aux valeurs maximale et minimale comme suit :

$$Sim(a, b, c) = 1 - \left| \frac{a - b}{Max_c - Min_c} \right|$$

Où a et b sont des valeurs numériques et c est un concept. Max_c et Min_c sont des valeurs maximal respectivement minimal pour le concept c .

3. Un module d'appariement de prix et de créneau horaire :

En faisant correspondre le prix et le créneau horaire, le SDA tente de rechercher des fournisseurs avec des services Cloud qui ont des créneaux horaires disponibles qui correspondent aux créneaux horaires spécifiés des consommateurs et avec une petite différence de prix entre les prix acceptables du fournisseur et du consommateur. Une fonction d'utilité U est utilisée pour évaluer le niveau d'appariement pour chaque appariement entre les prix et les horaires respectifs d'un consommateur et d'un fournisseur. L'utilité de U est donnée comme suit :

$$U = \alpha \cdot U^P(P_{min}^{prov}) + \beta \cdot U^T(T_s^{prov})$$

Où α et β sont la préférence d'un consommateur pour des prix de service et des créneaux horaires plus adaptés, et $\alpha + \beta = 1$. $U^P(P_{min}^{prov})$ est une utilité de prix à la consommation définie comme suit :

$$U^P(P_{min}^{prov}) = (P_{max}^{cons} - P_{min}^{cons}) / P_{max}^{cons}$$

Où P_{max}^{cons} est le prix maximum acceptable pour un consommateur, et P_{min}^{cons} est le prix minimum acceptable pour un fournisseur.

$U^T(T_s^{prov})$ est l'utilitaire de plage horaire d'un consommateur défini comme suit :

$$U^T(T_s^{prov}) = (T_s^{cons} \cap T_s^{prov}) / T_s^{cons}$$

Où T_s^{cons} est la plage de créneaux horaires possibles que le consommateur s'attend à réserver un créneau horaire pour utiliser un service, T_s^{prov} est la plage de créneaux horaires pendant lesquels un service est disponible.

4. Un module d'évaluation de service :

À l'aide de ce module, le service fourni par chaque fournisseur est évalué en fonction de la similitude entre les spécifications fonctionnelles et techniques du consommateur et du fournisseur. Le résultat consiste en une liste de services Cloud ordonnés en fonction des similitudes d'adéquation entre les contraintes fonctionnelles, techniques et budgétaires spécifiées par le consommateur et les fournisseurs de services.

3. Ontologie Cloud :

Dans ce travail, l'ontologie Cloud est constituée de plus de 400 concepts. Les services Cloud sont généralement classés en :

- Infrastructure en tant que service (IaaS) qui fournit des ressources de calcul (par exemple, l'EC2 d'Amazon fournit des VM).
- Données en tant que service (DaaS) qui permet aux utilisateurs de stocker des données sur des disques distants (par exemple, le S3 d'Amazon).
- Logiciel en tant que service (SaaS) qui fournit un logiciel spécialisé, accessible à distance via Internet (par exemple, le CRM de Salesforce).
- Plate-forme en tant que service (PaaS) qui fournit aux développeurs Cloud des environnements de programmation (par exemple, Google App Engine fournit un environnement d'exécution Python et une API permettant aux applications d'interagir avec l'environnement d'exécution Cloud de Google).
- Communication en tant que service (CaaS) fournit une bande passante dédiée, la sécurité du réseau, le cryptage et la surveillance du réseau.

5. Crawlers Cloud :

Chaque Crawler Cloud parcourt le www pour extraire des pages Web pertinentes pour les services Cloud, après il collecte des informations pour chaque site Web des services Cloud de fournisseurs, ensuite, il examine le contenu et extrait les données de lien (composées d'URL et d'hyperliens) et le contenu de la page Web. À l'aide d'une ontologie de concepts Cloud, le Crawler filtre le contenu des pages Web et détermine si le contenu est pertinent pour les services Cloud. Un document qui contient plus de concepts Cloud reçoit un score plus élevé. Il examine les documents pertinents en extrayant le nom du fournisseur de services, le type de service, le prix et les spécifications techniques telles que la vitesse du processeur et la capacité de la RAM . Toutes ces informations ainsi que l' URL de la page Web seront stockées dans la base de données de Cloudle.

4. Base de données de services Cloud :

Les fournisseurs de services Cloud enregistrent leurs services dans la base de données des services Cloud. De plus, les Crawlers Cloud sont également utilisés pour construire et maintenir la base de données Cloudle à partir des pages Web de fournisseurs.

A.6 Solution de Han et al [47]

Taekgyeong Han et Kwang Mong Sim ont mis en place un système (CSDS) pour découvrir et trouver le meilleur service Cloud. Comme illustrée à la figure (A.35). Ce système est composé de :

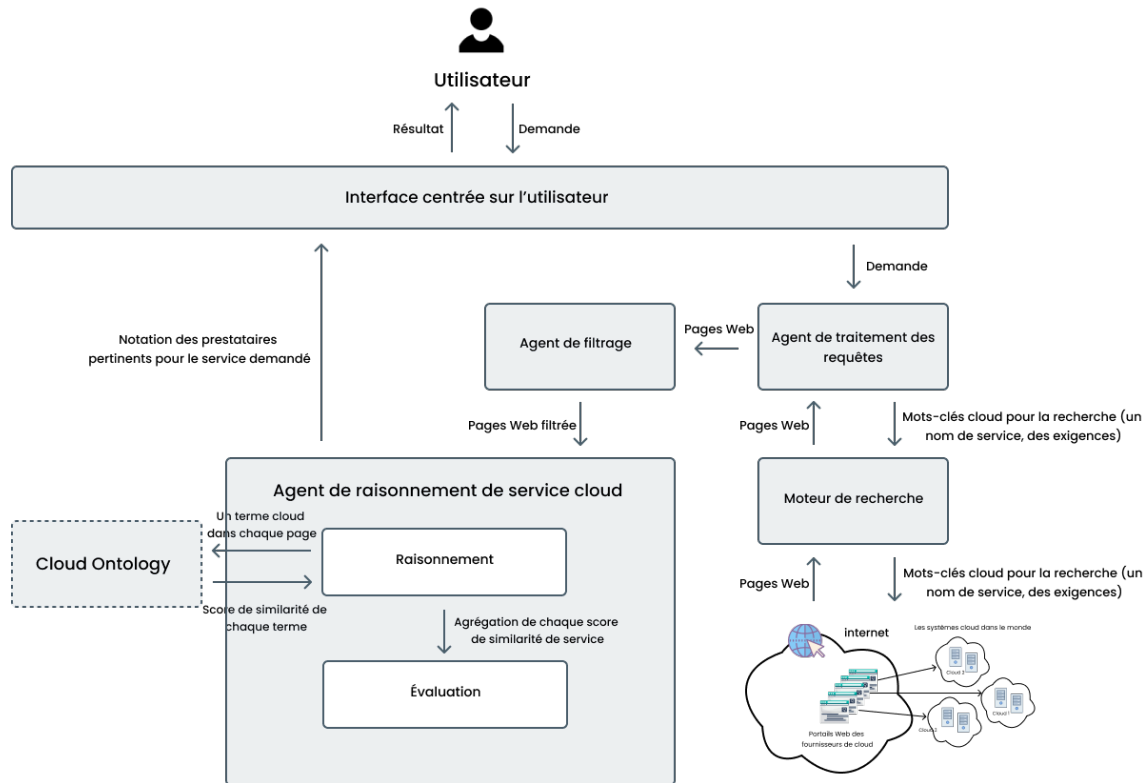


FIG. A.33 : Architecture de cadre selon la solution de Han et al [47]

1. Moteur de recherche :

Le CSDS dans ce travail utilise un moteur de recherche existant (par exemple Google) comme mécanisme de recherche initial pour collecter des informations sur les sites Web de services Cloud de fournisseurs.

2. Les agents :

Ce système est basé sur les trois agents suivants :

- **Agent de traitement des requêtes (QPA) :**

Le QPA localise les sources d'information en exécutant un moteur de recherche classique comme Google. Si le nombre de résultats de recherche est inférieur au nombre spécifié par l'utilisateur, une nouvelle requête alternative sera générée pour plus de résultats.

- **Agent de filtrage :**

Il remplace l'utilisateur pour visiter, consulter et filtrer tous les sites Web au cours du processus de recherche d'informations, en tenant compte des relations tel que les synonymes, les hyponymes et les hyperonymes à partir de WORDNET. La pertinence des pages web est déterminée en adoptant 3 heuristiques :

1. **Détection des expressions de preuve (EP) :**

L'agent de filtrage recherche des expressions de preuve (EP) telles que des mots-clés exacts, des synonymes, des hyponymes et des hypernymes pour les mots-clés de la requête d'utilisateur sur une page Web après avoir supprimé tous les mots fonctionnels (par exemple, "et" et "de") sur cette page et dans la requête d'utilisateur. Nous initialisons d'abord la valeur EP à zéro, puis pour chaque mot de la requête de l'utilisateur (sans les mots de fonction), si ce mot est dans l'ensemble de mots de la page Web, nous incrémentons la valeur EP de 1, sinon si son synonyme est présent, nous augmentons par 0.8, dans le cas des hyponymes, nous augmentons par 0.6, et pour les hyperonymes, nous augmentons par 0.4. Au final, nous divisons la valeur finale de l'EP par le nombre total de mots sur la page Web.

2. **Compter les fréquences de EP :**

L'agent de filtrage favorise les sites Web où les phrases de preuve se produisent plus fréquemment. Par exemple si le mot « science » apparaît fréquemment sur une page Web, il semblerait raisonnable de croire que cette page contient des informations sur la « science ». L'agent de filtrage détermine la fréquence d'occurrence des expressions de preuve (EP) pour chaque mot-clé dans une requête d'utilisateur sur un site Web. La métrique de fréquence EFL dans une page Web

est déterminée par le nombre total de correspondances exactes, de synonymes, d'hyponymes et d'hypernymes pour chacun des mots clés non fonctionnels de requête d'utilisateur.

3. Voir de la proximité entre les mots-clés :

En tenant compte de la proximité, l'importance potentielle des informations récupérées est susceptible d'augmenter. Si la requête est "London symphony orchestra", et si "London", "symphony" et "orchestra" apparaissent de manière adjacente dans une page Web particulière P, P est plus susceptible de contenir des informations pertinentes que lorsque les mots sont séparés par d'autres mots. Combinaison des 3 heuristiques : Un agent de filtrage évalue la pertinence d'une page Web comme suit :

$$Pertinence = 0,34 * EP + 0,33 * EF + 0.33 * proximit.$$

- **Agent de raisonnement du service Cloud (CSRA) :**

Il effectue deux fonctions :

1. **Raisonnement :** CSRA consulte l'ontologie Cloud pour effectuer un raisonnement de service. Toutes les informations fournies par l'utilisateur sont utilisées pour déterminer la similitude entre les services. Il existe trois méthodes pour déterminer la similarité : le raisonnement de similarité, le raisonnement équivalent et le raisonnement numérique. Dans l'article ils ont présenté que le premier raisonnement.

- **Le raisonnement de similarité :** Le raisonnement de similarité consiste à calculer la similarité entre deux concepts en calculant les super noeuds communs accessibles. Il peut être déterminé comme indiqué dans l'équation (A.3).

2. **Évaluation :**

L'utilité du service est utilisée pour déterminer l'évaluation. La page Web qui présente l'utilité du service le plus élevé sera sélectionnée comme le meilleur service pour l'utilisateur.

3. Ontologie Cloud :

L'ontologie Cloud dans ce travail représente les relations entre les services Cloud pour faciliter le raisonnement du CSRA sur les relations entre et parmi les concepts de services Cloud. Il comprend des concepts de services Cloud qui sont actuellement utilisés et de nombreux services qui pourraient sortir dans un proche avenir.

- **Service IaaS** fournit du matériel, des logiciels et des équipements pour fournir des environnements d'applications logicielles avec un modèle de tarification basé sur l'utilisation des ressources.
- **Service PaaS** offre un environnement intégré de haut niveau pour créer, tester et déployer des applications personnalisées.
- **Service SaaS** fournit un logiciel spécialisé accessible à distance par les consommateurs via Internet avec un modèle de tarification basé sur l'utilisation.

4. Interface utilisateur :

Lorsque le CSDS est déployé, un certain nombre de fournisseurs Cloud sont générés pendant que chaque fournisseur publie environ 25 de ses services. Au total, environ 13 000 pages Web existaient dans le www virtuel.

- **Étape01:** L'écran de la figure (A.34) montre la requête d'entrée de l'utilisateur qui contient un nom de service (par exemple, "Visual_Studio_2010") et des exigences comme le Os, CPU Name, horloge CPU, etc.

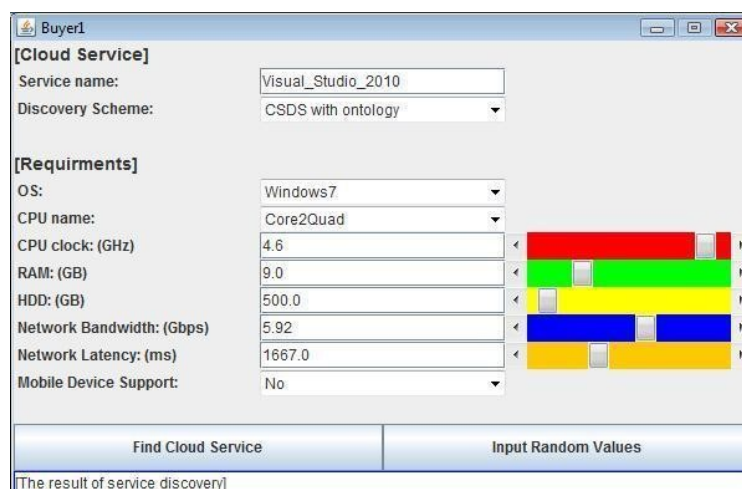


FIG. A.34 : Interface utilisateur CSDS [47]

- **Étape02:** Le CSDS recherche automatiquement avec un nom du service, "Visual Studio 2010" du virtual-www et filtrer les pages Web qui n'incluent pas le terme « Cloud ».
- **Étape03:** Le CSDS consulte l'ontologie Cloud pour le raisonnement des services. Ensuite, la similarité de chaque terme est agrégée en tant que utilitaire de service.
- **Étape04:** Le CSDS prend l'utilité la plus élevée, comme le meilleur service parmi 53pages Web.
- **Étape05:** Le CSDS renvoie le résultat de la découverte de service et le meilleur service.

A.7 Solution de Hasan et al [48]

Le cadre de cette solution est divisé en six sous-systèmes et dix composants. Comme le montre la figure (A.35).

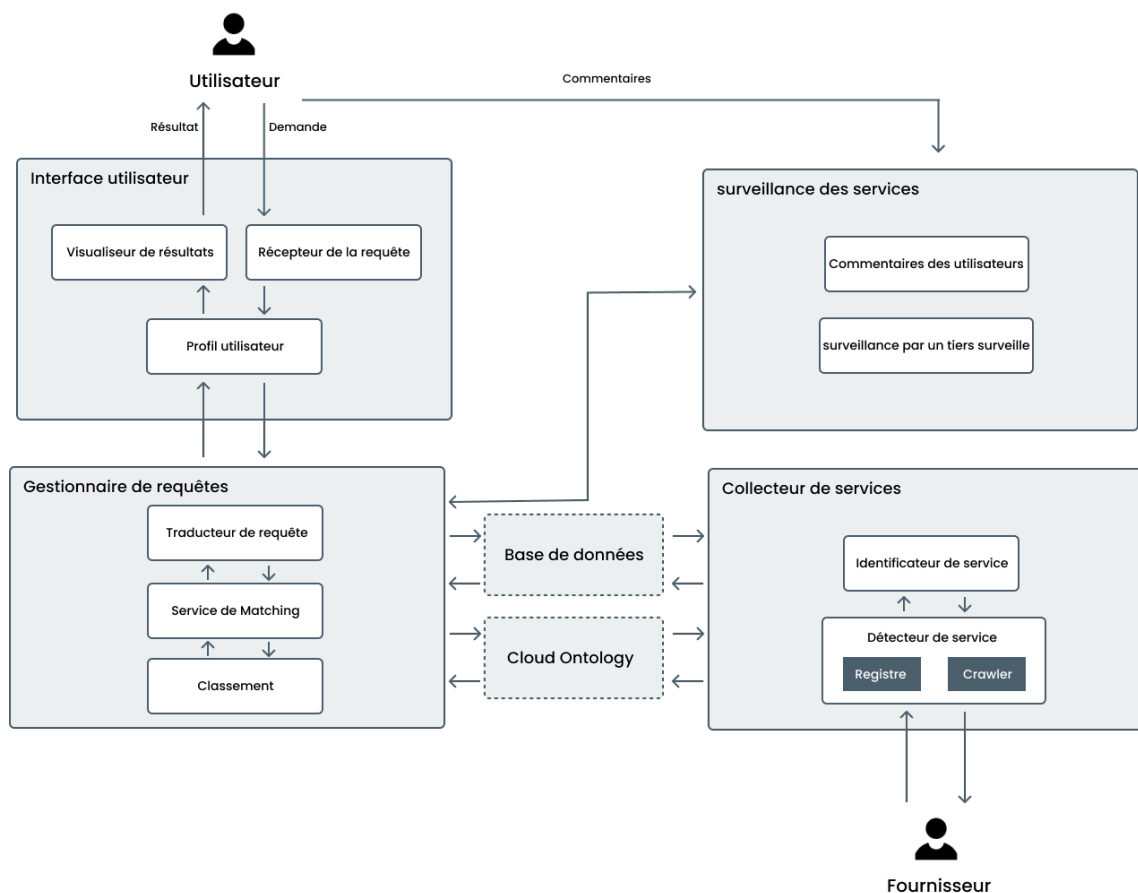


FIG. A.35 : Architecture de cadre selon la solution de Hasan et al [48]

1. Sous-système d'interface utilisateur :

C'est une interface graphique qui facilite la communication entre l'utilisateur final et le système. L'interface utilisateur contient trois composants comme suit :

1.1. Récepteur de la requête : Il reçoit les requêtes des utilisateurs dans divers formats. L'utilisateur saisit des requêtes en langage naturel ou utilise des listes prédéfinies, des cases à cocher ou des boutons radio pour saisir des requêtes.

1.2. Profil de l'utilisateur : C'est l'intermédiaire entre le sous-système d'interface utilisateur et le gestionnaire de requêtes.

1.3. Visualiseur de résultats : Il affiche les résultats correspondants pour l'utilisateur sous forme de liste ordonnée. L'utilisateur peut modifier les préférences de classement pour voir l'ordre différent des services correspondants.

2. Sous-système du gestionnaire de requêtes :

Le sous-système de gestionnaire de requêtes reçoit les requêtes des utilisateurs et renvoie une liste classée des services correspondants au utilisateur. Il contient trois composants comme suit :

2.1. Traducteur de requête : Il reçoit les requêtes de l'utilisateur pour extraire la requête sémantique basée sur l'ontologie Cloud. Ce traducteur peut utiliser l'approche de traitement du langage naturel (NLP) pour convertir une requête en texte brut ou une requête prédéfinie en une requête sémantique.

2.2. Service de Matching : Le composant Service Matching contient l'algorithme et l'approche que le système appliquera pour trouver le meilleur service pour le consommateur de Cloud.

2.3. Classement des services : Le composant de classement des services classe les services correspondants en fonction des préférences de l'utilisateur.

3. Sous-système d'ontologie Cloud :

L'ontologie Cloud facilite le raisonnement sémantique entre la demande de l'utilisateur et les services Cloud disponibles en fournissant une compréhension partagée du domaine des

services Cloud. Ce travail a construit une ontologie de domaine de service Cloud basée sur NIST et des informations collectées à partir des sites Web des fournisseurs de services Cloud.

4. Sous-système de collecteur de services :

Le collecteur de services collecte les services Cloud publiées par les fournisseurs de services Cloud dans divers formats. Le sous-système de service du collecteur comprend les deux composants suivants :

4.1. Identificateur de service : L'identificateur de service classe et catégorise les services Cloud découverts en fonction de différentes méthodes et techniques.

4.2. Détecteur de service : Le service détecteur collecte les descriptions de services Cloud avec différents formats. Généralement, il existe deux approches pour la détection de service :

- La première est le moteur de recherche sur Crawler où les fournisseurs de Cloud annoncent leurs services sur des sites Web sans aucune communication avec CSDS.
- La seconde est l'approche de registre où les fournisseurs de services Cloud doivent enregistrer leurs services via une communication directe avec CSDS.

5. Sous-système de Base de données :

Le sous-système collecteur de services maintient un référentiel de services à jour. Ce référentiel contient toutes les descriptions de services Cloud disponibles avec leur représentation sémantique.

6. Sous-système de surveillance des services :

Le sous-système de surveillance des services garantit que le service Cloud respecte l'accord de niveau de service (SLA) et fournit les commentaires au sous-système du gestionnaire de requêtes.

Service de surveillance se compose de deux composants comme suit :

6.1. Commentaires des utilisateurs : Recueille les commentaires des utilisateurs sur les performances des services Cloud.

6.2. Surveillance par un tiers : Surveille les fournisseurs de services Cloud pour s'assurer que le fournisseur de Cloud respecte les SLA.

Similitude sémantique basée sur l'ontologie Cloud :

La similarité sémantique détermine à quel point un concept A est lié au concept B. Cette article propose deux approches pour calculer la similarité entre deux concepts :

- **Approche par attributs dominants et récessifs :**

Les attributs du service Cloud sont divisés en attributs maîtres (dominants) et attributs esclaves (récessifs). L'existence de tous les attributs dominants est nécessaire pour accepter la service Cloud comme alternative et l'absence d'un seul attribut dominant suffit à rejeter le service Cloud. De l'autre d'autre part, l'existence ou l'absence des attributs récessifs n'est affectant le score de correspondance de l'alternative au service Cloud.

– La similarité des attributs récessifs est calculée sur la base de l'équation :

$$PDSim(x, y) = \begin{cases} 1 - \frac{|x-y|}{x}, & y < 2x \\ 0, & y \geq 2x \end{cases} \quad (A.4)$$

Si $y < 2x$ alors y est similaire à x et la valeur de similarité est $PDSim(x, y)$.

Si $y \geq 2x$ alors la distance entre les attributs est grande et la similarité est nulle, donc le consommateur de Cloud doit changer la requête pour obtenir résultats différents.

– La similarité entre deux valeurs d'attribut dominantes (y, z) est calculé comme suit :

$$DSim(y, z) = y \wedge z$$

$x, y \in 0, 1$: 0 et 1 représentent respectivement l'absence ou l'existence de l'attribut.

– Le score de correspondance entre la demande de l'utilisateur et l'alternative de service Cloud est le produit de la similarité totale de tous les attributs dominants $DSim$ et le pourcentage total de similarité de distance de tous les attributs $PDSim$ comme suit :

$$ms = \prod_{i=1}^v DSim(cs^{ai}, ur^{ai}) * \frac{\sum_{j=1}^u PDSim(cs^{aj}, ur^{aj})}{u}$$

v est le nombre d'attributs dominants et u est le nombre des attributs récessifs.

Si $ms > th$, le service Cloud est accepté comme alternative. tel que th est le seuil correspondant entrée par l'utilisateur.

- **Similitude sémantique basée sur l'approche ontologique :**

Cette approche divise les attributs de service Cloud en deux types : numérique et non numérique.

- Similitude des attributs numériques a été calculé sur la base de l'algorithme de similarité de distance en pourcentage équation (A.4).
- La similarité des attributs non numériques a été calculée comme montré dans l'équation (A.3).
- Le correspondante entre la demande de l'utilisateur et l'alternative au service Cloud est en moyenne de le pourcentage total de similarité de distance $PDSim$ de tous les attributs numériques et la similarité sémantique totale SQ de tous les attributs non numériques attributs comme suit :

$$ms = \left(\frac{\sum_{j=1}^u PDSim(cs^{aj}, ur^{aj})}{u} + \frac{\sum_{i=1}^v SQSim(cs^{ai}, ur^{ai})}{v} \right) / 2$$

u est le nombre d'attributs numériques et v est le nombre d'attributs non numériques. Si $ms > th$, le service Cloud est accepté comme alternative.

La figure (A.36) ci-dessous montre les résultats du marché des services Cloud pour un utilisateur qui a demandé un service DaaS avec les paramètres suivants (Vcpu = 4, Ram = 10 Go, Storage = 75 Go, Availability = 99%, price = 30 USD/month, Location = Inde, OS = Win, Backup = oui et $th = 0,9$).

User Request							
		4	10	75	99	30	
Cloud Service Marketplace							
Execution Time= 26							
Score Mean= 0.9307744107744108							
Score Threshold= 0.9							
Id	ProviderName	Vcpu	Ram GB	Storage GB	Availability %	Price \$	Score
76	test3	4	12	75	100	35	0.962
26	greenhousedata	4	8	100	99	35	0.93
47	1	4	10	75	99	20	0.9
Total DaaS Services		70	Total Matched Services		3	All Services	187

FIG. A.36 : Interface de service Cloud disponible sur le marché [48]

A.8 Solution de Beheshti et al [49]

La figure (A.37) illustre les principaux composants du moteur de recherche de services Cloud (CSSE), qui se compose de six couches principales à savoir :

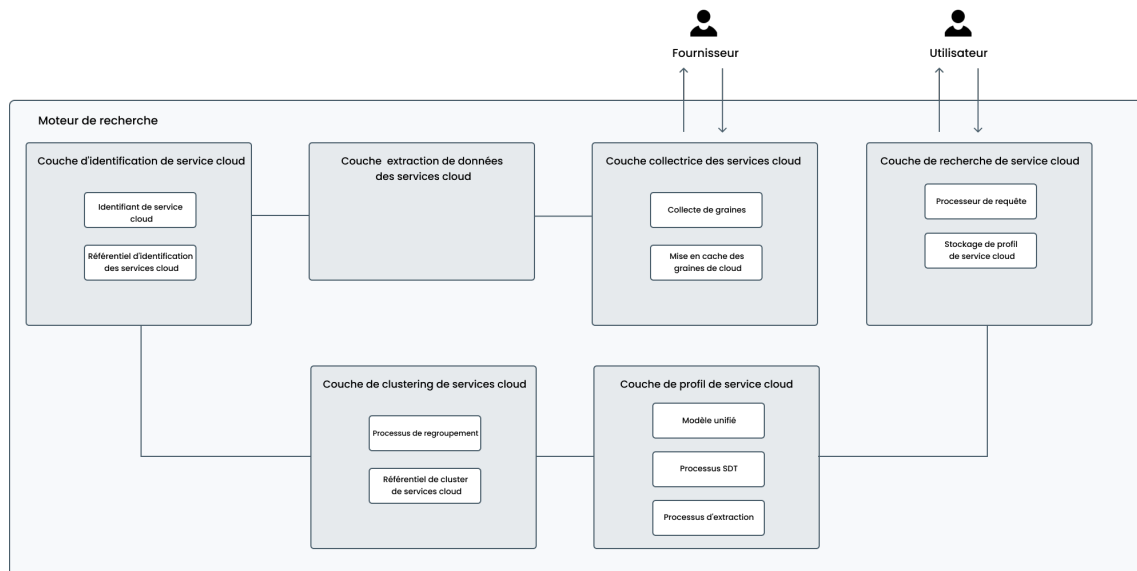


FIG. A.37 : Architecture de cadre selon la solution de Beheshti et al [49]

1. Couche collectrice des services Cloud :

Cette couche est responsable de la collecte des graines de services Cloud possibles (URL des services Cloud) dans des environnements réels. pour collecter les graines de service Cloud en utilise deux approches. Tout d'abord, nous développons le module de collecte de source de services Cloud qui est capable de collecter automatiquement les services Cloud en explorant les portails Web et les index sur les moteurs de recherche. Deuxièmement, nous développons le module basé sur l'interrogation des graines de service Cloud qui a la capacité d'interroger un service Cloud et de déterminer si ce service Cloud a été mis en cache dans le référentiel de graines de service Cloud. Cette demande peut être effectuée à la fois par les clients de services Cloud et les fournisseurs de services Cloud. De plus, si les utilisateurs du service Cloud demandent une graine enregistrée dans le système, le système peut renvoyer directement le résultat de la demande. Sinon, la graine sera envoyée à la couche de données d'extraction des services Cloud pour obtenir les détails essentiels qui peuvent déterminer si la graine fournit le service Cloud.

2. Couche de données d'extraction des services Cloud :

Cette couche est responsable de l'extraction du contenu essentiel dans la source du service Cloud, comme la description, les mots-clés, le contenu textuel et les hyperliens. Le contenu des services Cloud peut être obtenu automatiquement en filtrant la source du service Cloud. Ensuite, le contenu des services Cloud est envoyé à la couche d'identification des services Cloud, tandis que les sources des hyperliens des services Cloud peuvent être envoyées à la couche de cluster de services Cloud.

3. Couche d'identification des services Cloud :

Cette couche est responsable de l'identification du fournisseur de services Cloud. L'identifiant Cloud Services contient le processus d'identification des fonctionnalités pour déterminer si une source donnée est un service Cloud ou non. Ce traitement s'appuie sur la méthode de classification pour réaliser l'identification. L'identification peut être mise à jour automatiquement après l'identification d'un nouveau fournisseur de services Cloud pour améliorer les connaissances d'identification.

4. Couche de clustering de services Cloud :

Cette couche est responsable de la mise en cluster des fournisseurs de services Cloud. Le clustering de services Cloud est capable de collecter le service Cloud le plus similaire dans des clusters basés sur la méthode de clustering. Les clusters sont construits en fonction de deux fonctionnalités qui sont le texte des services Cloud et les hyperliens des services Cloud.

5. Couche utilisateur du moteur de recherche :

Cette couche fournit une interface Web permettant aux utilisateurs de rechercher des services Cloud. Un utilisateur peut simplement spécifier un mot-clé de recherche pour trouver des services Cloud. Elle peut également spécifier d'autres contraintes (par exemple, des catégories comme IaaS) pour affiner la portée de la recherche.

6. Couche de profil Cloud des services Cloud :

Cette couche est chargée de générer un profil de service Cloud basé sur les processus suivants :

6.1. La modélisation : Ce processus est responsable de la construction du modèle de service Cloud. Le modèle est construit sur la base des caractéristiques du service. De plus, nous observons plusieurs services Cloud pour identifier les caractéristiques du service telles que le type, le prix, la capacité. Ce modèle est utilisé pour trouver le service dans un cluster en enquêtant sur les fonctionnalités. Le processus commence par la sélection d'un fournisseur de services Cloud et détermine les fonctionnalités de service qui le fournissent. Ensuite, nous construisons un modèle JSON pour ce service.

6.2. Détection et suivi : Ce processus est responsable de la détection et du suivi d'autres services Cloud basés sur le modèle de fonctionnalité de service. Ce traitement peut rechercher le service à l'intérieur du cluster en prenant le modèle JSON qui est construit pour le service et suivre ce modèle. Le processus de détection et de suivi peut enquêter sur l'ensemble des sites Web du service Cloud pour trouver le service.

6.3. Extraction et stockage : Ce processus est responsable de l'extraction et du stockage du modèle JSON du service Cloud. Les détails des services Cloud peuvent être reçus du processus de détection et de suivi. Ensuite, nous pouvons stocker ces détails dans le modèle JSON pour prendre en charge la création d'un moteur de recherche de services Cloud. Nous mettons à jour ce processus pour découvrir les différentes fonctionnalités du service Cloud.

Annexe B

Algorithmes

Algorithme 1 Raisonnement des services

Input : demande d'utilisateur, service Cloud de l'ontologie

Output : Similarité

```
1 Double Caractéristiques_communes=0.0;
2 Double Similarité=0.0;
3 if demande.Price = service.Price then
4   | Caractéristiques_commune++;
5 end
6 if demande.Ram = service.Ram then
7   | Caractéristiques_commune++;
8 end
9 if demande.Disk = service.Disk then
10  | Caractéristiques_commune++;
11 end
12 if demande.Bandwith = service.Bandwith then
13  | Caractéristiques_commune++;
14 end
15 if demande.CPU = service.CPU then
16  | Caractéristiques_commune++;
17 end
18 if demande.ECU = service.ECU then
19  | Caractéristiques_commune++;
20 end
21 if demande.OS = service.OS then
22  | Caractéristiques_commune++;
23 end
24 Similarité=Caractéristiques_commune/7;
25 return Similarité
```

Algorithm 2 Algorithm de Matching

Input : Domain Node, Set of provider FF keywords**Output** : HashMap of similarity of each FF in Domain Node

```
1 children= GetChildren(Node);
2 VisitedNode.add(Node); // to avoid the cyclic mode
3 foreach child ∈ children do
4   if (!VisitedNode.contains(child)) then
5     synonyms.addAll(BabelNet.Synonyms(child));
6     foreach key ∈ keywords do
7       | Similaritykeys=similaritykeys+diceCoefficient(key, child);
8     end
9     similaritySim = similaritykeys/ keywords.size();
10    foreach key ∈ keywords do
11      | synonymesKey.addAll(BabelNet.Synonyms(key));
12      | similaritylexical =similaritylexical+LexicalSimilarity(synonymesKey,synonyms);
13    end
14    similarityLex =similaritylexical /sysnsetTerm.size();
15    foreach key ∈ keywords do
16      | similarityStructure=WuPalmer(key, child);
17      | Sim.add(Str);
18    end
19    similarityStru = max(Sim);
20    P1=Math.exp(similaritySim);
21    P2 = Math.exp(similarityLex);
22    P3 = Math.exp(similarityStru);
23    Similarity = (P1* similaritySim+P2*similarityLex+P3*similarityStru)/(P1+P2+P3);
24    similarityChildren.put(child, Similarity);
25  end
26 end
27 } return similarityChildren;
```

Algorithme 3 Algorithme de Découverte

```
void BFS-basedMatchingKeywords(OntModel model, Node, EnglishTerms, BabelBased-
Dic, matchedChildren){
  children = getChildren(model, Node);
  SetNodeVisited(Node); // to avoid the cyclic mode
  /* children that achieve high matching degree (>60% for Unique keywords)
     against EnglishTerms(using relative entoropy) will be fetched      */
1 foreach child ∈ Children do
2   if (!VisitedNode.contains(child)) then
3     keywords=getKeywords(model,child,"Unique");           // if child has unique
       keywords
4     H_rel=RelativeEntropy(keywords, EnglishTerms, child, BabelBasedDic);
5     if (H_rel >= 0.6) then
6       matchedChildren.add(child);
7       keywords=getKeywords(model,child,"Merged");
8       H_rel = RelativeEntropy(keywords, EnglishTerms, child, BabelBasedDic);
9       if (H_rel >= 0.6) then
10        BFS-basedMatchingKeywords(model, child, EnglishTerms, BabelBasedDic,
        matchedChildren);
11      end
12    end
13  end
14 end
15 }
```

Algorithme 4 Calcul de similarité H_{rel}

Double RelativeEntropy(keywords, serviceSLATokens, FF, BabelBasedDic){

ArrayList<String>terms=keywords.split(","); // convert string to array

1 int index[]=new int[terms.size()];

2 **JSONObject** domainDic=BabelBasedDic.get(FF);3 **foreach** *term* \in *terms* **do**

4 | termCount=getCount(term, serviceSLATokens);

5 | synonym=TrackingSynonyms (term, serviceSLATokens, domainDic);

6 | **if** (synonym !=null) **then**

7 | | termCount+=getcount(synonym, serviceSLATokens);

8 | | index[i++]=termCount;

9 | | totalcount+=termCount;

10 | **end**11 **end**12 **For** (i=0; i<terms.size(); i++){13 **if** (index[i]>0) **then**

14 | | Pi=index[i]/totalcount;

15 | | Entropy+=(Pi * Log(Pi))/Log(terms.size());

16 **end**17 } **return-** Entropy;

18 }

Annexe C

Calcul de la similarité

Le calcul de la similarité entre les concepts est un problème de longue durée dans le domaine du traitement du langage naturel. Il fait référence à l'évaluation de la correspondance entre deux concepts provenant d'une base de connaissances. Il existe plusieurs façons d'évaluer la similitude entre les deux entités. On peut trouver :

1. Mesure terminologique :

Ces méthodes sont utilisées pour calculer la valeur de similarité des termes ou des chaînes de caractères ou bien les textes. Il existe deux approches pour cette mesure :

- **Approche syntaxique :**

Cette méthode analyse la structure des chaînes à comparer, plus la séquence de caractères dans la chaîne, le nombre de fois qu'une lettre apparaît dans la chaîne pour concevoir une mesure de similarité, plus ils partagent de caractères. En revanche, ils ne profitent pas du sens du terme. Parmi de telles mesures de similarité syntaxique On peut trouver :

- **Distance de Dice :**

La similarité de Dice est définie par le nombre des objets communs multipliés par 2 sur le nombre total d'objets. Elle est donc définie par la formule suivante :

$$S_{Dice}(S_1, S_2) = \frac{2|S_1 \cap S_2|}{|S_1| + |S_2|}$$

- **Distance de Jaccard :**

L'indice de Jaccard est le rapport entre la cardinalité (la taille) de l'intersection des ensembles considérés et la cardinalité de l'union des ensemble. Il

permet d'évaluer la similarité entre les ensembles. Soit deux ensembles S_1 et S_2 l'indice est :

$$J((S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

– **Distance de Levenshtein :**

La distance de Levenshtein calcule la similarité entre les représentations sous forme de chaînes de caractères. Elle est égale au nombre minimal de caractères qu'il faut supprimer, insérer ou remplacer pour passer d'une chaîne à l'autre.

Soit **Insert(M,P)** la somme des insertions nécessaire pour avoir les mêmes caractères entre le mot M et le mot P.

Soit **Supp(M,P)** la somme des suppressions nécessaire pour avoir les mêmes caractères entre le mot M et le mot P.

Soit **Permut(M,P)** la somme des permutations nécessaire pour avoir le même ordre des caractères dans M et P.

La distance de Levenshtein est définie comme suit :

$$Lev(M, P) = Insert(A, B) + Supp(A, B) + Permut(A, B)$$

Et pour avoir la similarité Syntaxique :

$$SimSyn(M, P) = 1\left(\frac{Lev}{\max(|M|, |P|)}\right)$$

• **Approche lexicale :**

La similarité lexicale est la mesure du degré de ressemblance entre des séries de mots en utilisant des ressources externes comme Wordnet. Cette similarité est calculée à partir des liens sémantiques déjà existants dans les ressources externes. Pour le calcul de la similarité, la fonction $Syn(c)$ calcul l'ensemble des synsets de wordnet du concept c , soit $S = Syn(c1) \cap Syn(c2)$ l'ensemble des sens communs entre $c1$ et $c2$ à comparer, la cardinalité de S est :

$$Card(S) = |Syn(c1) \cap Syn(c2)|$$

Soit $\min(|Syn(c1)|, |Syn(c2)|)$ le minimum entre la cardinalité des deux ensembles $Syn(c1)$ et $Syn(c2)$ alors la mesure de similarité lexicale entre les deux concepts $c1$ et $c2$ est définit comme suit :

$$SimLex(c1, c2) = \frac{Card(S)}{\min(|Syn(c1)|, |Syn(c2)|)}$$

2. Mesure structurelle :

Ce sont les méthodes qui déduisent la similarité entre deux entités en exploitant leurs positions dans une hiérarchie et en fonction des informations structurelles. En effet, les entités sont reliées entre elles par des liens sémantiques ou syntaxiques. Cette mesure sera calculer par :

- **La mesure wu et palmer (Wup) :**

La métrique de similarité de Wu et Palmer mesure la profondeur de deux concepts donnés dans la taxonomie WordNet, et la profondeur de leur plus bas ancêtre commun (lowest common subsumer(LCS)) et les combine pour obtenir un score de similarité :

$$SimWup(c1, c2) = \frac{2 \cdot profondeur(LCS)}{profondeur(c1) + profondeur(c2)}$$

3. Mesure sémantique :

La similarité sémantique est une évaluation du lien sémantique afin d'estimer à quel degré deux concepts sont proches dans leurs sens, elle est calculée par la combinaison des similarités terminologique et structurelle, donc elle est définis selon la formule :

$$SimSem(c1, c2) = \frac{P1 \cdot SimSyn(c1, c2) + P2 \cdot SimLex(c1, c2) + P3 \cdot SimStruct(c1, c2)}{P1 + P2 + P3}$$

Tel que :

$$P1 = e^{SimSyn}$$

$$P2 = e^{SimLex}$$

$$P3 = e^{SimStruct}$$

Le résultat de cette mesure est compris entre 0 et 1, plus la valeur est proche de 1 plus les deux termes sont similaires.

Annexe D

Notions sur les ontologies

On commence par le web d'aujourd'hui. Le contenu du Web d'aujourd'hui est conçu pour des lecteurs humains, n'est pas fait pour être manipulé de façon intelligente par les programmes informatiques. Il provoque une perte de temps considérable avant d'obtenir les informations requises. Ainsi que les moteurs de recherche ne comprennent pas le sens du document, et sont encore très inefficaces malgré le nombre croissant de pages. Et donc le web sémantique est venu pour résoudre ce problème.

D.1 Définition du Web Sémantique

Le Web Sémantique n'est pas un Web séparé, mais bien un prolongement du Web que l'on connaît et dans lequel on attribue à l'information une signification clairement définie, ce qui permet aux ordinateurs et aux humains de travailler en plus étroite collaboration. Le but initial du Web Sémantique à l'origine c'est de permettre les échanges de données entre les agents aussi il permet aux ordinateurs de faire des recherches intelligentes.

D.2 Les Couches du Web Sémantique

Le développement du Web Sémantique s'opère par étape. Chaque étape construit une couche au dessus d'une autre. La figure D.1 suivante montre les principales couches du Web Sémantique :

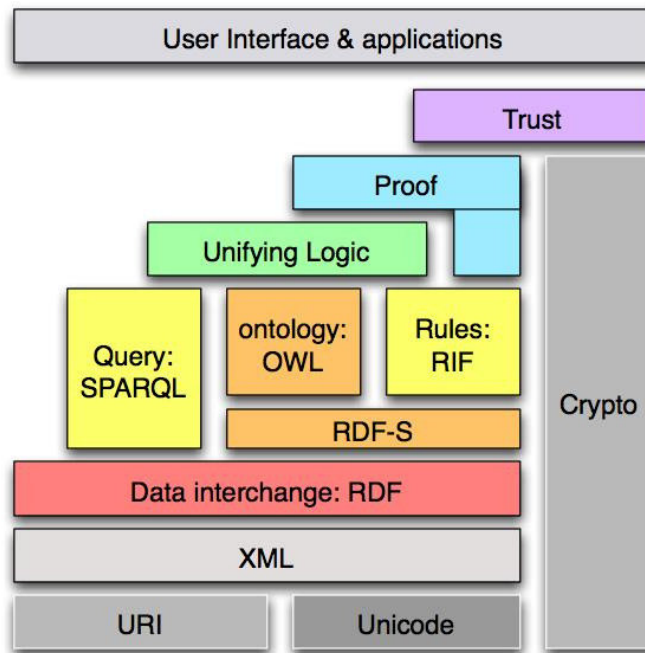


FIG. D.1 : Les principales couches du Web Sémantique

D.2.1 La couche URI

L'URI est un protocole permettant d'identifier toute ressource sur le web. Dans le cas du Web Sémantique, l'URI est une séquence de caractères avec une syntaxe restreinte, qui permet d'identifier toute ressource utilisée dans le cadre d'une application Web Sémantique.

D.2.2 La couche XML, XML Schema

Il s'agit d'une couche syntaxique, de bas niveaux, permettant de structurer les données selon un format de message standard qui est XML. La couche XML joue le même rôle que l'HTML dans le Web syntaxique, il hérite certaines de ses caractéristiques, .En d'autres termes, XML permet d'indiquer l'organisation logique de l'information d'un document, mais ne permet pas d'en décrire le contenu Il permet de structurer l'information et donc séparer la mise en forme et le contenu.

Avant de commencer à organiser les informations dans un document XML, il est impératif de définir la structure de ce dernier en utilisant XML-S. XML-S est un langage de description de format de document XML permettant de définir la structure et le type de contenu de ce dernier.

D.2.3 La couche RDF

RDF est un modèle de graphe destiné à décrire de façon formelle les ressources Web et leurs métadonnées (Une métadonnée est une donnée de haut niveau, c'est une donnée qui définit ou décrit une autre donnée), de façon à permettre le traitement automatique de telles descriptions.

Le but initial de RDF est une bonne représentation et une meilleure exploitation des métadonnées.

Un document structuré en RDF est un ensemble de triplets. Un triplet RDF est une association (Sujet, Objet, Prédicat) ;

- Le sujet représente la ressource à décrire, peuvent être référencé par un identificateur. Cet identificateur doit être une URI, par exemple une page Web.
- Le prédicat représente un type de propriété applicable à cette ressource.
- L'objet représente une donnée ou une autre ressource. L'objet peut être littéral (valeur) ou ressource.

Il existe plusieurs syntaxes pour décrire des ressources en RDF :

- **Abstraite** : Syntaxe sous forme de graphe permettant de représenter graphiquement la description de ressources
- **N triples** : Un graphe RDF est représenté par une collection de triples
- **Notation N3**: Représentation flexible et lisible de graphe rdf, elle utilise une notation similaire à la syntaxe N-triples elle permet d'utiliser les préfixes et elle permet de combiner plusieurs descriptions d'une seule ressource
- **XML** : C'est la notion la plus utilisée.

D.2.4 La couche RDF Schéma

RDFS ajoute à RDF la possibilité de définir des hiérarchies de classes et de propriétés dont l'applicabilité et le domaine de valeurs peuvent être contraintes à l'aide des attributs `rdfs:domain` et `rdfs:range`. À chaque domaine applicatif peut être ainsi associé un schéma identifié par un préfixe particulier et correspondant à une URI. Les ressources instances sont ensuite décrites en utilisant le vocabulaire donné par les classes définies dans ce schéma.

Pour résumer, XML peut être vu comme la couche de transport syntaxique, RDF comme un langage relationnel de base, RDFS offre des primitives de représentation de structures.

D.2.5 La couche Ontologique

Une ontologie est une spécification explicite et formelle d'une conceptualisation partagée. La conceptualisation est le résultat d'une analyse d'un domaine du monde réel, en élaborant à ce dernier un modèle abstrait, dont les concepts les relations et les contraintes sont explicitement définis dans un format et langage formel. Cette formalisation consiste à rendre cette conceptualisation exploitable par la machine.

Les raisons qui nécessitent le développement d'une ontologie :

- Partager la compréhension commune de la structure de l'information entre les personnes ou les fabricants de logiciels.
- Permettre la réutilisation du savoir du domaine.
- Expliciter les concepts utilisés et les contraintes sur un domaine.
- Interopérabilité : l'interopérabilité est une spécialisation de la communication qui permet de répertorier les concepts que des applications peuvent s'échanger même si elles sont distantes et développées sur des bases différentes.

D.2.5.1 Composants d'une ontologie

Une ontologie est composée de : concepts, propriétés, relations, axiomes et des instances.

Nous détaillons ci-dessous ces différents composants :

- **Les concepts** : il peut représenter un ensemble d'objet, une idée, un principe ou une notion abstraite.
- **Les propriétés** : sont des caractéristiques valuées attachées aux concepts. Permettent de relier des individus à des valeurs de données (Par exemple : Nom, Prenom, Adresse..)
- **Les relations** : elle exprime les associations entre les différents concepts. Permettent de relier des instances à d'autres instances. Ces relations incluent les relations de spécialisation-généralisation (sous classes), relations d'agrégation ou de composition (relation is-a, partie de), relations d'association et d'instanciation.
- **Les axiomes** : constituent des assertions considérées toujours comme vraies. Ils com-

binent des concepts, des relations pour définir des règles d'inférence et permettent de vérifier la validité des informations spécifiées ou de déduire de nouvelles informations.

- **Les instances** : elles constituent la définition extensionnelle d'une ontologie. Elle est utilisé pour représenter des éléments dans le domaine.

D.2.5.2 La couche ontologique

Les ontologies peuvent être classifiées selon quelques cote :

- **Ontologies d'application** : ont un domaine de validité restreint et correspondent l'exécution d'une tâche.
- **Ontologies de domaine** : ont un faisceau plus large, une bonne précision et ne sont pas propres à une tâche particulière. Ontologies propres aux télécommunications, à l'industrie minière, à la biologie génétique, le transport, la géographie, le génie logiciel, etc.
- **Ontologies générales** : ne sont pas propres à un domaine.
- **Ontologies supérieures** : représentent des concepts généraux comme l'espace, le temps ou la matière. Elles sont universelles

D.2.5.3 Langage d'ontologies

Pour décrire une ontologie, plusieurs langages ont proposés dans la littérature tels que : XML, RDF, RDFS et OWL.

- **Langage d'ontologie OWL** : est un langage de description d'ontologies conçu pour la publication et le partage d'ontologies sur le web sémantique. Il s'appuie sur RDFS afin de l'enrichir en définissant un vocabulaire plus complet pour la description d'ontologies complexes. Cette richesse ajoutée à RDFS se matérialise par l'ajout de nouvelles notions telles que : propriétés de classe équivalente, Symétrie entre les ressources, transitivité, cardinalité, etc.

Le OWL facilite l'interopérabilité au niveau machine du contenu du web plus que ce qui est déjà supporté par les XML, RDF, RDFS en fournissant du vocabulaire supplémentaire avec des sémantique formelles.

D.2.6 SPARQL

SPARQL pour Simple Protocol and RDF Query Language, est un langage de requêtes permet d'interroger les métadonnées sous forme de fichiers RDF, un protocole pour émettre et envoyer des requêtes SPARQL (services Web) vers des serveurs dédiés et en recevoir les résultats. C'est l'un des trois normes fondamentales du Web Sémantique, avec RDF et OWL.

D.2.7 La syntaxe de SPARQL

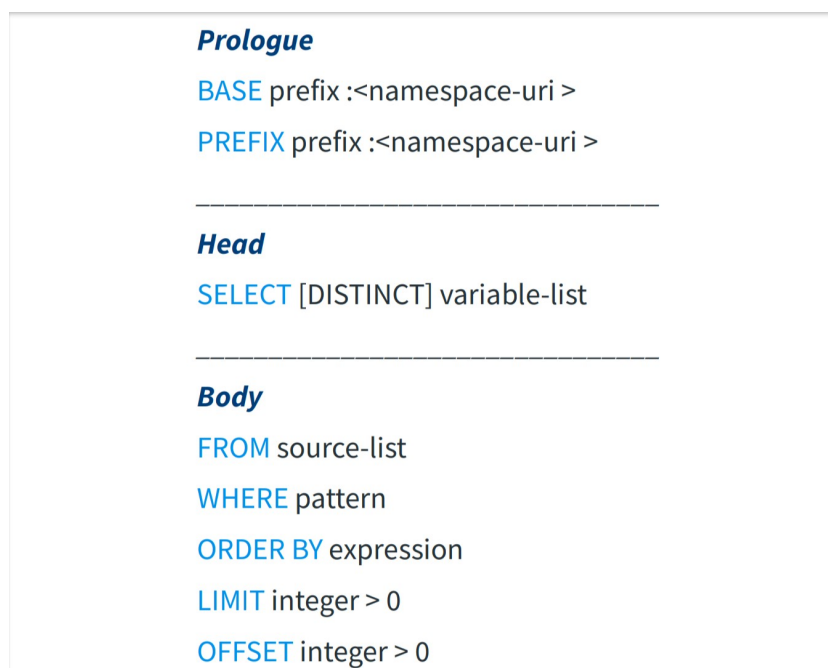


FIG. D.2 : Syntaxe d'une requete SPARQL

- La clause SELECT suivi d'une liste de variables dont on souhaite connaître les valeurs répondant à la requête.
- La clause FROM (optionnel) permet de spécifier différents graphes de triplets dans lesquels il faut chercher. Si pas mentionné, on suppose qu'un graphe a été spécifié au processeur SPARQL.
- La clause WHERE suivi d'une déclaration de pattern « sujet, predicat, objet » pouvant contenir plusieurs patterns, des filtres, des unions, des présences optionnelles.
- ORDER BY c'est à dire trier par..
- LIMIT pour donner un nombre maximal de réponses.
- OFFSET pour commencer à partir d'un numéro de réponse.

D.2.8 Les types de requêtes SPARQL

- **SELECT** : rechercher de ressources du modèle, résultats restitués sous un format tabulaire.
- **ASK** : indiquer si la requête retourne un résultat non vide, retourne (true or false).
- **DESCRIBE** : obtenir des informations à propos de ressources présentes dans le modèle.
- **CONSTRUCT** : construire de nouveaux graphes RDF à partir des résultats de la requête .