REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOHAMED KHIDER - BISKRA

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
جامعة محمد خيضر بسكرة

Faculté des sciences exactes et des sciences de la nature et de la vie
Département d'informatique

THÈSE

Pour l'obtention de grade de

DOCTEUR 3ème CYCLE EN INFORMATIQUE

Option : Techniques de l'image et de l'intelligence artificielle

Titre

Découverte de services web via le Cloud computing à base d'agents mobiles

Par

SAOULI Hamza

Soutenue le :

Devant le jury composé de

Président :	Professeur Chaoui Allaoua de l'université de Constantine 2
Rapporteur :	Professeur Kazar Okba de l'université de Biskra
Co-Rapporteur :	Dr. Benharkat Aïcha-Nabila, maître de conférences A de l'INSA de Lyon
Examinateur :	Dr. Terissa Labib Sadek, maître de conférences A de l'université de Biskra
Examinateur :	Dr. Bennoui Hammadi, maître de conférences A de l'université de Biskra
Invité :	Professeur Chikhi Salim de l'université de Constantine 2

Biskra, Algérie.

Remerciements

En premier lieu, je remercie le bon dieu de m'avoir donné la force et la patience nécessaire pour achever ce travail de thèse.

Je tiens à remercier KAZAR Okba, professeur à l'université de Biskra, de m'avoir accueillie au sein de son équipe pour réaliser ma thèse ainsi que pour sa disponibilité et son soutien. Je remercie également BENHARKAT Aïcha-Nabila, Docteur à l'INSA (Institut National des Sciences Appliquées) de Lyon, d'avoir co-encadré mon travail, de son suivi et ses conseils.

Je remercie les membres du jury de m'avoir fait l'honneur d'accepter de participer à mon jury de thèse.

Je tiens aussi à saluer toute ma promotion de doctorat et tous mes amis.

Enfin, je remercie tous ceux qui ont contribué de près ou de loin à l'aboutissement de ce travail de recherche.

ملخص

مع التطور السريع في الخدمات السحابية، في العدد والوظائف، الحاجة إلى محرك للاكتشاف و تركيب الخدمات اصبح ضرورة لا غنى عنها. لتحقيق مثل هذا النظام، نحن بحاجة إلى استخدام تقنيات مناسبة لطبيعة هذا النوع من ا التطبيقات، وهي تكنولوجيا الوكيل والحوسبة السحابية. تكنولوجيا الوكيل مناسبة جدا للنظم التي تتطلب التعاون بين عدة كيانات من أجل تحقيق هدف مشترك، مثل إنجاز نظم حيث يعمل الوكلاء معا للعثور على أفضل خدمات الإنترنت التي تلبي المتطلبات الوظيفية و الغير وظيفية للزبائن. وعلاوة على ذلك، فإن طبيعة توزيع الخدمات السحابية تتطلب منا استخدام الوكيل المتحرك للبحث مما يؤدي لخلق اتصالات آمنة وخفض التكاليف بين محرك البحث ومقدمي خدمة الإنترنت. تكنولوجيا الحوسبة السحابية، هي ميزة كبيرة لتطبيقات الإنترنت الموزعة. من جهة هذه التكنولوجيا تتيح للزبائن دفع ثمن ما يستخدمونه فقط من الموارد الافتراضية والبرمجيات ويجنبهم شراء النسخ الجديدة للنظام الذي يستخدمونه. من ناحية أخرى، هذه التكنولوجيا تتيح للزبائن الحصول على موارد بسعة تخزين غير محدودة. في هذه الأطروحة نهتم بدراسة واقتراح محرك للاكتشاف و التركيب يمكنه استيعاب ومعالجة الآلاف من الاستفسارات والملفات في وقت واحد، ومع أفضل وقت اجابة. في هذه الأطروحة نقترح بنية جديدة تعتمد على الحوسبة السحابية و الوكلاء لاكتشاف و تركيب خدمات SaaS. هذه البنية تدعم تنفيذ آلية جديدة للاختيار تركيب وتصنيف الخدمات ، وتستند هذه الآلية على قياس جديد يسمى ب "درجة الوجود"، هذا القياس يسمح لنا بحساب درجة التكيف بين الاستفسار و شجرة الملف WSDL. وبالإضافة إلى ذلك، فإن هذه الآلية تجمع المعايير الوظيفية و الغير وظيفية لطلب خدمات SaaS، والذي يسمح للخدمات المركبة بتلبية المتطلبات الوظيفية وغير الوظيفية للزبائن. أظهرت نتائج المحاكاة فعالية محرك الاكتشاف و التركيب خدمات SaaS خاصة زمن الاستجابة. وينصب التركيز على ضرورة تطوير النموذج الاقتصادي الذي يتكيف مع عمليات وتقنيات هذا النوع من المحركات. أخيرا، اقترحنا نموذج للمقارنة والتقييم من أجل مقارنة النهج المقترح مع انهج أخرى.

كلمات البحث: الحوسبة السحابية، إدارة الموارد السحابية ، نظام متعدد الوكيل ، الوكيل المتحرك ، خدمات الويب، البرمجيات كخدمة (SaaS)، الاكتشاف والاختيار، التركيب و الترتيب، WSDL، نموذج للمقارنة.

Résumé

Avec la rapidité de l'évolution des services Cloud, en nombre comme en fonctionalités, le besoin d'un moteur de découverte et de composition de services devient une nécessité incontournable. Afin de réaliser un tel systeme, nous devons faire recours aux technologies adaptées à la nature de ce type d'applications, à savoir, les technologies d'agents et cloud computing. La technologie d'agent est très adéquate pour la modélisation des systèmes qui nécessite une collaboration entre plusieurs entités, afin de réaliser un but commun, comme la réalisation d'un système où les agents collaborent pour trouver les meilleurs services web qui correspondent aux exigences fonctionnelles et non-fonctionnelles des clients. Outre, la nature distribuée des services Cloud nous impose l'utilisation d'agents mobiles afin de chercher et créer des pistes de communications sécurisées et à moindre coût entre le moteur de recherche et les fournisseurs de services web. La technologie cloud computing, représente un atout considérable pour les applications distribuées sur internet. D'un côté cette technologie permet aux clients de payer seulement ce qu'ils utilisent comme ressources virtuelles et logicielles et leurs évite d'acheter à chaque fois la nouvelle version du système qu'ils utilisent ; d'un autre côté, cette technologie permet d'avoir un support physique et virtuel avec des capacités de stockage et de traitement illimitées. Dans notre thèse on s'est à l'étude et la proposition d'un moteur de découverte et composition qui peut accueillir et traiter des milliers de requêtes et de fichiers WSDL en même temps, et avec les meilleurs temps de réponse. Dans cette thèse nous proposant une nouvelle architecture Cloud computing basée agents pour la découverte et composition des services SaaS. Cette architecture a pour but de supporter et d'exécuter un nouveau mécanisme de sélection, composition et classification de services, ce mécanisme est basé sur une nouvelle mesure, qu'on a appelé "le degré d'existence", cette mesure nous permet de calculer le degré d'adaptation de la requête par rapport aux représentations arborescente des fichiers WSDL. De plus, ce mécanisme combine les paramètres QoS et fonctionnels pour effectuer la composition de services SaaS, ce qui permet d'avoir des services composites qui répondent aux exigences fonctionnelles et non fonctionnelles des clients. Les résultats de la simulation montrent l'efficacité du moteur de découverte et composition de services SaaS notamment en temps de réponse. L'accent est mis sur la nécessité de développer un modèle économique qui s'adapte aux fonctionnements et techniques de ce genre de moteur. Enfin, nous avons proposé un modèle de comparaison et d'évaluation afin de pouvoir comparer l'approche proposée avec les autres approches du domaine.

Mots-clés : Cloud Computing, Gestion de ressources Cloud, Système Multi-Agent, Agent mobile, Service web, Softaware-as-a-Service (SaaS)**,** Découverte et Sélection, Composition et Triés, WSDL, Modèle de Comparaison.

Abstract

With the rapid evolution of cloud services, in number and functionality, the need for a discovery and service composition engine becomes an unavoidable necessity. To achieve such a system, we need to use appropriate technologies to the nature of this type of application, namely, agents and cloud computing technologies. The agent technology is very suitable for modeling systems that require collaboration between several entities in order to achieve a common goal, such as performing a system where agents work together to find the best web services that meet the functional and non-functional client requirements. Moreover, the distributed nature of cloud services requires the use of mobile agents to search and create secure communications between the search engine and the web service providers and to reduce costs. The cloud computing technology has a significant advantage for the distributed Internet applications. On the one hand, this technology allows clients to pay for virtual and software resources that they use and avoid buying the new versions of the cloud system that they use. On the other hand, this technology enables using physical and virtual ressources with unlimited storage and treatment capacities. In our work we are interested to study and propose a discovery and composition engine which has the capacities to treate thousands of queries and WSDL files simultaneously with the best response time. In this thesis we propose a new Cloud computing architecture based-agent for SaaS services discovery and composition. This architecture is intended to support and implement a new selection, composition and ranking mechanism. This mechanism is based on a new measure, so-called "existence degree", this measure allows us to calculate the degree of adaptation between the client query and the tree representations of WSDL files. In addition, the discevery and composition mechanisms combine the functional, QoS and SaaS services parameters to create composite services that meet the functional and non functional client requirements. The simulation results show the effectiveness of the discovery and composition engine especially for response time. The focus is on the need to develop an economic model which is adapted to the operations and techniques of this kind of engine. Finally, we proposed a comparison and evaluation model in order to compare the proposed approach with the other approaches from the same domain.

Keywords: Cloud Computing, Cloud Resource Management, Multi-Agent System, Mobile Agent, Web Service, Softaware-as-a-Service (SaaS), Discovery and Selection, Composition and Ranking, WSDL, Comparison Model.

Table des matières

Liste des figures

Liste des tableaux

Liste des algorithmes

Terminologie

Application

Une application est un logiciel créé par un client ou un fournisseur de services. Dans la couche Software-as-a-Service une application est également appelée service. Une application est destinée à être créée et déployée sur une machine virtuelle. Par la suite le client n'a qu'a installé l'image de son application sur les machines virtuelles cibles pour qu'elle soit utilisée à chaque fois qu'on invoque une de ces machines virtuelles.

Arbre WSDL

Un arbre WSDL est la représentation structurelle de son fichier WSDL. Un arbre WSDL sert comme moyen pour faciliter le matching structurel entre la requête du client et les informations contenues sur les fichiers WSDL. Il est également très recommandé d'utiliser ces arbres pour effectuer la composition entre services sélectionnés, en utilisant les différents modèles de coûts d'édition entre arbres.

Chaine de connectivité

Une chaine de connectivité est une collection de services ordonnés, ou les inputs du service précédent correspondent aux outputs du service suivant. La notion de chaine de connectivité est utilisée pour exprimer les différentes possibilités de composition de services parmi un ensemble de services sélectionnés ou stocké sur une base de services.

Un client est celui qui demande un service complexe d'un fournisseur de service, afin de le consommer. Un client peut être un simple utilisateur final, ou une entreprise qui utilise le service demandé pour des fins commerciales ou de développement. Le client est destiné également à payer l'abonnement ou la facture d'utilisation du service.

Degré d'existence

Le degré d'existence est une nouvelle mesure qu'on propose pour le calcul du degré de correspondance entre la requête du client et les arbres WSDL. A l'encontre du degré de similarité, le degré d'existence permet de préserver la requête et les arbres WSDL sous leurs formes originales, ce qui nous évite de perdre une partie de la sémantique de ces deux derniers.

Fournisseur de service

Dans l'architecture Cloud, un fournisseur de service est une entreprise ou un particulier qui présente des services sur Internet. Ces services sont destinés à être consommés par les clients. Ces services peuvent être des applications, des machines virtuelles, des adresses IP, connexion réseau… etc.

Instance

Une instance est une copie d'une application ou d'une partie d'une application. Une instance peut être lancée ou arrêtée par son service d'hébergement. Chaque application peut être constituée d'une ou plusieurs instances (applications distribuées).

Machine Virtuelle (MV)

Une machine virtuelle est un système d'exploitation isolé par son hyperviseur physique hôte. Le client exécute ces taches sur les machines virtuelles comme s'il était sur une machine hôte physique. Une MV est généralement crée à partir de plusieurs paramètres: un hôte configuré qui émule un hôte physique avec CPU virtuel, RAM, et disques de stockage virtuels; et une image qui donne un aperçu du système exploitation. L'image peut être équipée d'un logiciel supplémentaire pour fournir des différentes fonctionnalités.

Nœud Cloud

Un nœud Cloud est un ensemble de ressources Cloud regroupées ou distribuées sur différents sites et région Internet mais gérées par un même superviseur. Un nœud Cloud peut être un serveur Cloud, une armoire de serveur Cloud, ou tout un Datacenter.

Paramètres de Service

Tout service a besoin d'un modèle de description, qui utilise un ensemble de paramètres ou attributs pour décrire ses aspects : fonctionnels, non-fonctionnels, et Cloud.

Service

Dans un environnement Cloud, un service est un mécanisme qui permet l'accès aux capacités offertes par les fournisseurs, où l'accès est fourni à l'aide d'une interface web. Le service est généralement tune application ou une entité accessible par le public ou un groupe d'utilisateurs et qui permet de fournir des fonctionnalités logicielles particulières.

Service Atomique

Un service atomique est un service qu'on ne peut pas décomposer en plusieurs entités de services.

Service Composite

Un service Composite, dit aussi service complexe, est un service constitué de plusieurs services afin de créer de nouvelles fonctionnalités.

I
Introduction générale

I.1. Contexte du travail

La découverte, la sélection, la composition et le tri de services, représentent les opérations de base pour tout système de manipulation de services sur Internet. Ces opérations reposent essentiellement sur des techniques et des technologies Informatique et linguistique développées récemment. Un système de découverte et de composition repose sur trois éléments :

1. La requête du client : qui peut être saisie sous forme textuelle ou sous forme de formulaires remplis par l'utilisateur pour décrire les caractéristiques des services qu'il souhaite invoquer.
2. Le système intermédiaire : qui a pour rôle de récevoir la requête, de l'analyser, chercher et composer les meilleurs services qui correspondent aux caractéristiques décrites par le client.
3. Le fournisseur de services : c'est celui qui héberge et offre des services sur Internet.

La découverte est le processus qui permet de recueillir les services qui correspondent aux sous-requêtes du client qui sont éparpillés sur internet. La sélection consiste à choisir les meilleurs services qui correspondent aux sous-requêtes du client parmi ceux découverts. La composition consiste à présenter les services sélectionnés sous forme d'un seul service virtuellement composé et qui correspond à un maximum de sous-requêtes. Enfin, le tri consiste à classer les services composites ou atomiques sur la base d'un indice de classification qu'on appelle degré de matching et sert à aider le client à choisir le service qu'il désire utiliser.

Dans ce qui suit nous présentons, la problématique, les motivations et les objectifs du travail, quelques concepts généraux, les contributions, et enfin un guide de lecture de la thèse.

I.2. Problématique et Objectifs

Avec la croissance exponentielle en nombre et en fonctionnalités des services sur Internet, ainsi que la diversité des technologies utilisés pour l'implémentation et la présentation de ces derniers, il est devenu essentiel de proposer un système de découverte et de composition de services qui permet au client de choisir les meilleurs services qu'il désire sans avoir à effectuer n'importe quelle tâche manuellement.

La mise en œuvre d'un système de découverte et composition de services nécessite cinq principaux éléments :

1. Un algorithme de matching pour la sélection de services : un tel algorithme va permettre au système de découverte et composition de sélectionner les meilleurs services qui correspondent aux exigences fonctionnelles du client.
2. Un algorithme de matching pour la composition de services : un tel algorithme va permettre au système de découverte et composition de créer des services

virtuellement composés, afin de présenter les services atomiques sélectionnés précédemment sous forme d'un seul service regroupant un maximum d'exigences fonctionnelles et non-fonctionnelles.

3. La modélisation des composants du système : pour le bon fonctionnement du système de découverte, il est nécessaire de bien modéliser les principaux composants du système en représentant chacun par un module ou un agent spécifique afin d'augmenter la robustesse du système.
4. La Gestion des ressources logiques et virtuelles du système : tout système de découverte et de composition doit avoir sa propre infrastructure d'exécution, qui permet d'offrir les ressources logiques (application) et virtuelles nécessaires pour l'exécution des algorithmes de matching. Ces ressources doivent être gérées intelligemment et efficacement pour réduire le temps de réponse et les coûts d'utilisation du moteur.
5. La liaison des ressources distribuées sur le réseau : afin d'assurer un fonctionnement optimal du système de découverte et composition. Il très important de relier tous les nœuds et Datacenter sur lesquels les algorithmes de matching sont déployés afin que tout sous-système qui a besoin de nouvelles ressources puisse accéder et utiliser les ressources des autres sous-systèmes du moteur de découverte et composition.

Beaucoup de travaux on été proposés pour résoudre le problème de découverte et composition de services (chapitre 3) mais aucun d'eux n'est arrivé à proposer un système complet qui regroupe les Cinq éléments décrit ci-dessus (chapitre 7).

L'objectif de cette thèse est la proposition d'un moteur de découverte, sélection, composition et de tri de Software-as-a-Service représenté sous forme de services web. Le moteur proposé doit également bénéficier des capacités offertes par la technologie Cloud et les possibilités de modélisation offertes par le paradigme agent pour implémenter et gérer ses propres ressources.

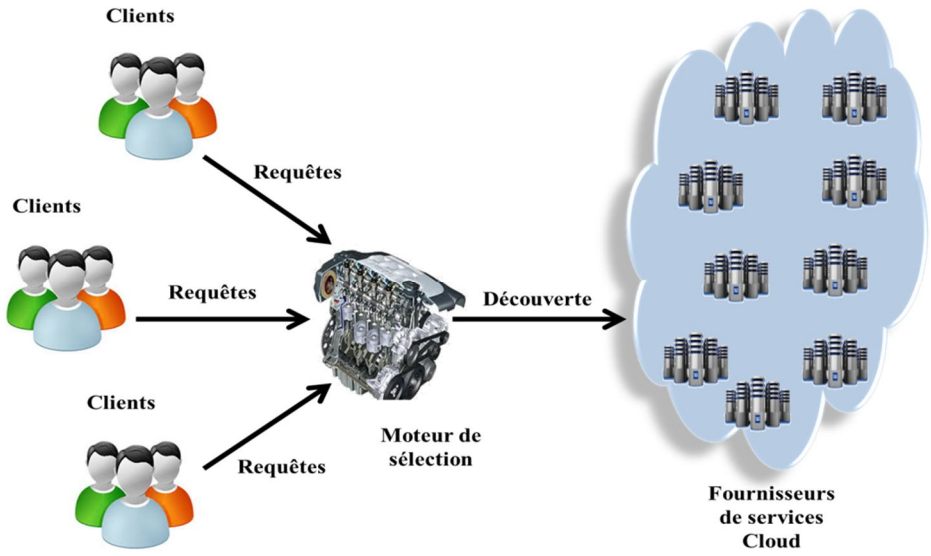

Figure I.1: Illustration du problème.

I.3. Concepts généraux

Dans cette section nous résumons les conceptes de base utilisés pour la proposition du moteur de decouvèrte et composition de services SaaS.

I.3.1. Concept d'Agent

Le concept d'agent a été essentiellement inspiré des domaines de philosophie et de la psychologie. Plusieurs définitions ont été données à travers les années, où chaque définition traite un ou plusieurs aspects de ce paradigme :

Selon [1], un agent est un système capable d'action autonome et réfléchie dans un environnement réel.

Selon [2], un agent est un logiciel persistant qui a un but bien précis, l'agent peut être distingué d'un logiciel classique par sa taille car plus petite et par ces objectifs et agendas sur les-quelles il se base pour accomplir ses taches.

Selon [3], un agent est un système informatique qui se trouve dans un environnement complexe et dynamique, et qui aperçoit puis réagit de façon autonome, afin de réaliser les buts pour lesquels il a été créé.

Selon IBM [5], un agent intelligent est un logiciel qui effectue un ensemble de tâches prédéterminées, avec un certain degré d'indépendance et d'autonomie, en se basant sur un ensemble de connaissances et une représentation d'objectifs prédéterminés.

Un agent peut être également, distingué d'autres entités logicielles par son [6]:

- Autonomie: un agent peut agir sans revenir à son possesseur afin de prendre des décisions.
- Comportement social: un agent a la faculté de communiquer et d'interagir avec les autres éléments de son environnement.
- Réactivité: un agent peut apercevoir et ensuite réagir aux différents évènements sur son environnement.
- Pro-activité: un agent a la faculté de prendre des initiatives afin de s'adapter au changement de son environnement.
- Persistance: un agent doit suivre son but sans interruption jusqu'à l'achèvement de ce dernier.
- Raisonnement et rationalité: un agent a la faculté de raisonner rationnellement afin de choisir les meilleures actions à entreprendre pour optimiser sa productivité.
- Mobilité: un cas particulier d'agent logiciel est l'agent mobile qui a la faculté de se déplacer d'une machine à une autre, afin d'exécuter des tâches de natures distribuées.

I.3.2. Système Multi-Agent (SMA)

Un système multi-agent est un système constitué d'un ensemble d'agents coopérants pour atteindre un but global ou un ensemble de buts distincts. En se basant sur des techniques de coordination et de négociation, les agents peuvent orchestrer leurs actions et arriver à des

consensus. Ceci permet d'avoir un comportement cohérent qui optimise la productivité des solutions émergentes. Les SMAs sont généralement utilisés pour représenter des systèmes où: les composants opèrent avec des informations incomplètes sur l'environnement (chaque composant peut être représenté par un agent), le contrôle des évènements et l'accès aux données sont décentralisés, et la communication est asynchrone [7].

Les SMAs son célèbres pour leurs faculté à représenter et à s'adapter aux problèmes réels et de nature complexe. Ils le sont également grâce aux performances et caractéristiques que l'on y trouve [8]:

- Fiabilité: la distribution des tâches sur des agents spécifiques, facilite la résolution des défaillances d'un système rapidement et efficacement.
- Extensibilité : l'Indépendance des agents les uns vis à vis des autres facilite la modification de leurs comportements.
- Robustesse: la coopération entre agents permet au système de faire face à des situations incertaines et d'augmenter la tolérance aux fautes, vu que l'échec d'un agent n'a pas d'influence sur le fonctionnement des autres.
- Maintenabilité: L'inter-indépendance entre agents permet également de maintenir chaque agent séparément des autres sans affecter le fonctionnement global du système.
- Evolutivité et Flexibilité: la faculté d'auto-adaptabilité permet aux développeurs d'ajouter ou de supprimer de nouvelles contraintes (ou même de nouveaux agents) au SMA sans pour autant altérer le mécanisme global du système.
- Efficacité : la capacité de communication entre agents permet de développer des systèmes de calcul distribués très puissants et très efficaces.
- Réduction des coûts: un système centralisé est toujours gourmand en temps de développement et de maintenance, avec les SMA on peut décentraliser la gestion et l'exécution des tâches, en créant des sous-systèmes représentés par des agents spécifiques, ce qui réduit les coûts d'extension et de maintenance.

I.3.3. Agent mobile

Un agent mobile est un logiciel autonome et auto-adaptable qui a la faculté de migrer d'une machine à une autre et de suspendre ou changer son comportement selon les évènements et les circonstances qu'il rencontre.

La définition de Gray et al nous semble la plus distinctive [9]:

"A mobile agent is an executing program that can migrate, at times of its own choosing, from machine to machine in a heterogeneous network. On each machine, the agent interacts with stationary service agents and other resources to accomplish its task".

Kalchuk et Karmouch ont étendu cette définition en ajoutant: *"The agent decides when and where it will migrate, and may interrupt its own execution and continue elsewhere on the network"* [10].

En général un agent mobile est composé : d'un code (statique), d'un état et de données. Lors de la migration d'un agent mobile d'une plate-forme à une autre, ses données et son état peuvent changer. On distingue deux types de mobilités: une forte mobilité, qui consiste à

transférer le code, les données ainsi que l'état de l'agent, et une faible mobilité qui consiste à transférer le code et les données seulement [11].

Les propriétés des agents mobiles, les placent à la tête de la liste des meilleures technologies de communication et de déploiement automatique [12]:

- Asynchronisme: un agent mobile n'a pas besoin de rester en permanent contact avec son propriétaire. L'agent mobile encapsule la tâche dont il est responsable, ensuite il migre vers la plateforme cible pour exécuter cette dernière, tout en coupant le contact avec sa plateforme d'origine, afin de dégager cette dernière de toute responsabilité de suivi ou de gestion. Après l'accomplissement de sa tâche l'agent rétablit à nouveau le contact avec sa plateforme si elle est disponible. Si elle ne l'est pas, il suspend son fonctionnement en attendant le rétablissement de la connexion.
- Autonomie: un agent mobile n'a pas besoin de revenir vers son propriétaire pour prendre des décisions.
- Auto-adaptabilité: un agent mobile a la faculté d'observer et d'adapter dynamiquement son comportement par rapport au changement qu'il aperçoit, afin de répondre à la contrainte qu'il rencontre.
- Réduction des coûts de communication: avec les agents mobiles, on n'a plus besoin de transférer d'énormes quantités de données d'un serveur a un autre, il suffit seulement d'envoyer un agent mobile avec le code nécessaire pour traiter les données sur le serveur cible, ce qui permet de réduire la consommation de la bande passante.
- Encapsulation du Protocole: avec la croissance rapide du nombre de serveurs déployés sur Internet, il devient très difficile de mettre à jour ou de maintenir les protocoles de communication entre eux. Un agent mobile encapsule en lui le Protocole de communication ce qui facilite l'automatisation du décodage des données par la machine réceptrice.
- Tolérance aux fautes: la capacité des agents mobiles à réagir dynamiquement aux situations défavorables, permet de construire des systèmes robustes et tolérants aux fautes. Si la plateforme tombe en panne, les agents mobiles migrent automatiquement vers d'autres plateformes et continuent leurs exécutions de façon normale.
- Adaptation aux réseaux hétérogènes: puisque les agents mobiles sont indépendants par rapport à la couche transport et ne dépendent que de leur environnement d'exécution, ils sont bien adaptés à tous genre de machine ou de plateforme.
- Contrôle en temps réel : il est très difficile voire impossible de gérer, en temps réel, des entités logicielles à distance vu les latences réseau qu'engendre ce type de contrôle. La technologie d'agent mobile représente un atout considérable pour résoudre ce problème, vu quelle permet de transférer le contrôle des entités, sur les serveurs cibles.
- Modèle de conception adapté: la nature distribuée des agents mobiles permet la conception et la construction rapide des systèmes distribués très fiables, puisque ce dernier est la raison d'être de la technologie d'agent mobile.

I.3.4. Services web

Selon Gartner research (June 15, 2001) [13, 14] les services web sont des composants logiciels faiblement couplés délivrés sur la base des standards d'Internet.

En général, les services webs sont des applications auto-descriptives et modulaires qui permettent d'offrir des services sur Internet sur la base d'un ensemble de Protocoles et de standards XML pour la description, publication et invocation de services à travers le globe [15].

L'utilisation de langages issus de la famille XML telle que WSDL ou SOAP pour la description et l'accès aux services web permettent d'implémenter ces derniers indépendamment de tout langage ou plateformes de développement. Ce qui facilite leurs découvertes et compositions en utilisant n'importe quel moteur ou Protocole de communication.

L'utilisation des services web comme interface d'interaction, d'échange de données et de représentation de tout genre de services sur Internet (tell que des services Cloud) engendrent des bénéfices divers [14]:

- Facilite, simplifie et standardise l'accès aux données par n'importe quel type de clients de n'importe quel endroit au monde en utilisant n'importe quel genre d'appareil.
- Facilite la communication intra et inter-entreprises.
- Offre un cadre simple mais efficace pour le développement d'application B2B vu l'utilisation des standards de communication les plus connus sur Internet pour effectuer des connexions entre applications web.
- Permet l'utilisation de courtiers (broker en anglais) pour faciliter l'accès et l'invocation dynamique de services.
- Facilite l'intégration de nouveaux services avec ceux déjà existants sur les réseaux afin de créer des services composites pour répondre à des requêtes complexes.
- Enfin, les services web représentent un atout considérable pour la représentation des services Cloud, vu les diverses ressemblances entre les services web en tant qu'application web et les Software-as-a-Service en tant qu'application en ligne, représentées par leurs Entrées/Sorties.

I.4. Contributions

Par rapport aux défis et problèmes vus dans la section précédente, la première et principale contribution est la proposition d'un modèle de découverte et composition de service basé sur le Cloud qui utilise la technologie d'agents comme outil de modélisation et d'implémentation des composants du moteur.

La seconde contribution est la proposition d'un modèle de description de Software-a-Service qui prend en considération non seulement les aspects communs entre services web et services Cloud mais aussi l'aspect Cloud des services SaaS.

La troisième contribution est la proposition d'un algorithme de matching qui met en œuvre la sélection, composition et tri de services SaaS, tout en prenant en considération toutes les caractéristiques et attributs descriptifs des services SaaS.

La quatrième contribution est la proposition d'un ensemble de systèmes de gestion de ressources logiques et virtuelles qui prennent en compte : la gestion des machines virtuelles, la gestion des unités de traitement (CPU), gestion d'espace de stockage, déploiement automatique des composants et la prédiction des besoins en ressources.

La cinquième contribution est la proposition d'un système d'agents mobiles qui assurent le recueil de services SaaS sur Internet, la mise-à-jour des services SaaS, la liaison des ressources et des nœuds du moteur proposé, et le transport des codes et données nécessaires pour le déploiement des composants Cloud.

La dernière contribution est la proposition d'un modèle de comparaison entre services SaaS qui permet de classer et d'effectuer une étude comparative entre approches de découverte et composition de services dans le Cloud.

I.5. Structure de la thèse

La thèse est structurée comme indiqué sur la figure I.2:

Le chapitre 1 présente le contexte et la problématique de la recherche. Ce chapitre identifie également les objectifs de la thèse et présente brièvement les principales contributions.

Le chapitre 2 présente un état de l'art complet sur le Cloud computing. Il présente une vue d'ensemble sur les contributions du Cloud computing en milieu universitaire et industriel, il clarifie la nature du Cloud, ses caractéristiques, et ce que le monde des technologies de l'information attend de ce nouveau concept. Ensuite, nous y présentons les différents types de Cloud ainsi que ces principaux défis.

Le chapitre 3 présente les concepts de découvertes et composition de services web ainsi que les différentes classifications et de catégorisations d'approches proposées dans la littérature. Ce chapitre présente également 49 travaux relatifs connexes qui seront catégorisés en plusieurs classes.

Le chapitre 4 aborde la notion de découverte et composition de services sur le plan matching. Au début ce chapitre, nous présentons le modèle de description de services SaaS. Ensuite nous présentons l'algorithme de sélection ainsi que sa liaison avec l'algorithme de composition. Enfin, nous présentons l'algorithme de composition et de tri de services SaaS ainsi que la manière de combiner des attributs descriptifs pour avoir des résultats de composition optimaux.

Le chapitre 5 aborde la notion de découverte et composition de services sur le plan architectural. Au début ce chapitre, nous présentons les composants de l'architecture proposée. Ensuite, nous présentons le fonctionnement et les interactions générales entres les différents composants. Enfin, nous présentons les principaux comportements d'agents ainsi que les mécanismes de gestion de ressources virtuelles et logiques proposés dans ce travail.

Le chapitre 6 décrit les outils d'implémentation, les paramètres de simulation et quelques tests empiriques de notre prototype.

Le chapitre 7 décrit en détail le modèle de comparaison proposé. Au début nous décrivons la taxonomie proposée, puis il décrit les différentes classes d'approches ainsi que les critères adoptés par ce modèle pour la comparaison entre systèmes de découverte. Ensuite, on rappelle brièvement les travaux relatifs à l'approche proposée en décrivant leurs inconvénients. Enfin, nous présentons deux tableaux comparatifs entre notre approche et les travaux connexes qui sont présentés.

Finalement, le chapitre 8 conclut cette thèse. Il synthétise les contributions globales et met en évidence les perspectives de cette recherche.

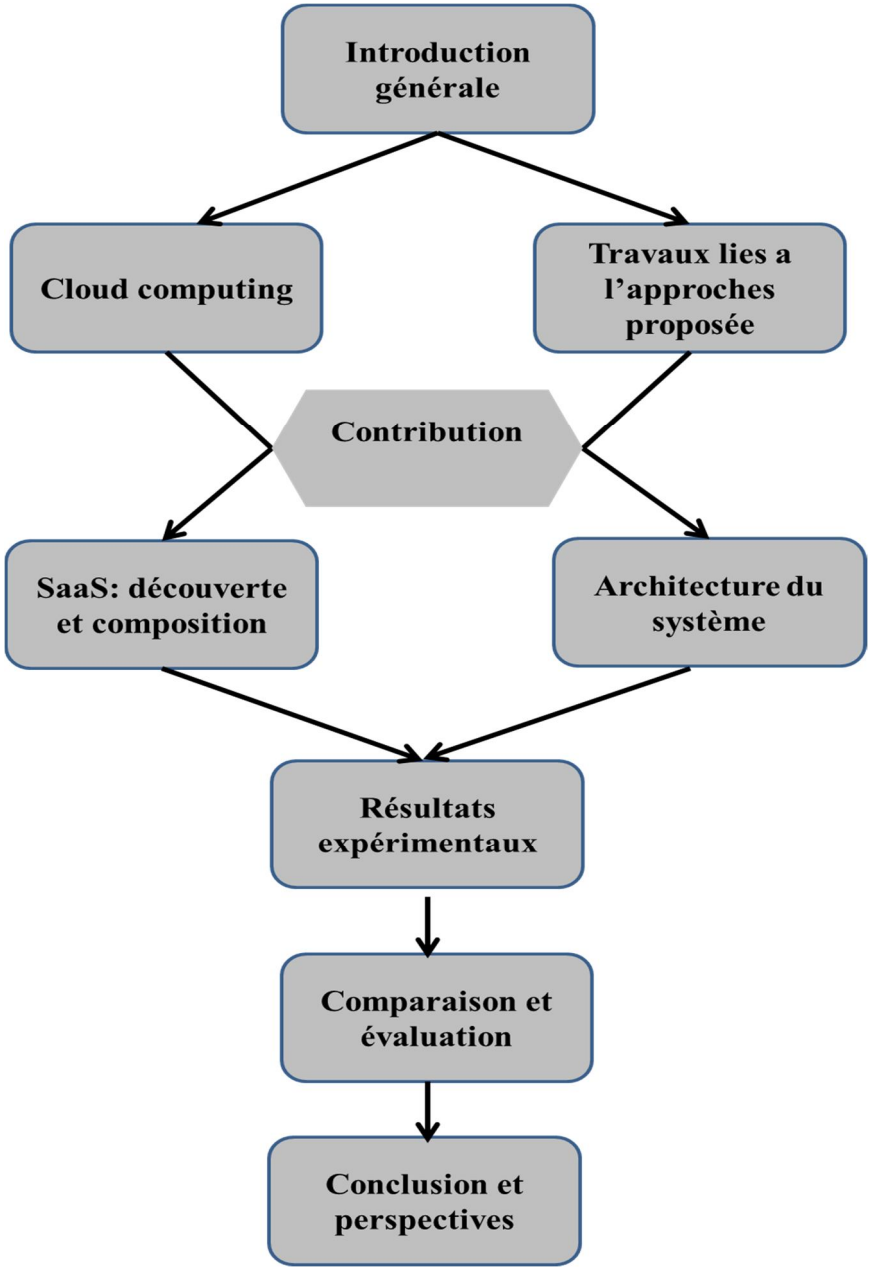

Figure I.2: Structure de la thèse.

Partie I. Etat de l’art et travaux connexes

Chapitre II: Cloud computing.

> *Comment est ont arrivé à la technologie Cloud? C’est quoi le Cloud? Quel est l’impact du Cloud sur la société? Quels sont les modèles sous-jacents derrière le succès du Cloud? Pourquoi doit ont adopté le Cloud? Quelles sont les limitations et points forts du Cloud? Comment contrôlé et accédé au Cloud? Quels sont les défis de la technologie Cloud?*

Chapitre III : Travaux liés à l’approche proposée.

C’est quoi la découverte de services? Pourquoi la découverte de service? Quelles sont les catégories d’approches de découverte? Quelle approche choisir? Quels sont les principaux aspects des mécanismes de découverte? C’est quoi la composition de services? Pourquoi la composition? De quelle manière la requête du client peut être satisfaite par un système de composition? Quelles sont les principales catégories d’approches de composition de services? Quelles sont les principales classes des travaux connexes? Quels sont les principaux apports des travaux connexes?

II
Cloud computing

II.1. Historique

Afin de comprendre la nature de la technologie Cloud computing (ou informatique nuagique) il est nécessaire de suivre l'évolution de l'informatique dès ses premiers pas vers une maturité concrétisée par le concept du Cloud.

L'évolution vers le Cloud a connu plusieurs étapes, qui ont changé la façon dont on conçoit nos logiciels dont on offre nos services, à travers l'internet:

1. L'ère des ordinateurs centraux: largement utiliser dans les années 60 et 70. Ces ordinateurs son dotés d'une grande capacité de traitement et de mémoire vive, et sont reliés a des périphériques d'entrées/sorties qui permettent aux développeurs d'exécuter leurs programmes, ce genre d'ordinateurs est destiné aux grands organismes telle que : les banques, les compagnies aériennes, les Universités, … etc.
2. L'ère des mini-ordinateurs:A partir de 1980 les organisations spécialisées dans la conception et développement des ordinateurs ont commencé à produire des ordinateurs de tailles de plus en plus petites par rapport aux ordinateurs centraux destinés aux opérations de niveaux départementals et administratifs, ce genre d'ordinateurs est appelé de nos jours serveurs.
3. L'ère des ordinateurs personnels (micro-ordinateur ou ordinateur individuel). Ce genre d'ordinateur est apparu en 1985, il est destiné à l'usage personnel avec une capacité de traitement et stockage plus petite que les mini-ordinateurs, ce type d'ordinateur a été conçu pour faciliter l'accès et le traitement d'informations pour le personnel administratif et pour les développeurs à partir de leurs bureaux de travail.
4. L'avènement du World Wide Web: en 1990 le britannique Sir. Timothy John Berners-Lee a proposé le concept du World Wide Web ou toile étendue dans le monde entier, ce concept est basé sur des liens hypertextes qui permettent de se déplacer d'une page web vers une autre par un simple clic. La navigation sur Internet est basée sur une architecture clients/serveurs qui consiste à stocker les pages web et les bases de données dans les serveurs de destination et par la suite le client accède à ces pages via un navigateur web (Ex. Mozilla Firefox, Google chrome, Internet explorer, … etc.)
5. Les appareils mobiles: bien que connus sous plusieurs formes auparavant, le mobile computing à commencer de prendre le devant vers l'an 2001, avec l'avènement des communications mobiles.Avec les appareils mobiles, et les logiciels mobiles les utilisateurs d'Internet peuvent transporter leurs données et applications dans leurs poches et accéder a leurs comptes sur Internet (compte bancaire, réseaux sociaux; … etc.)
6. Vers les architectures Cloud computing**:** la technologie Cloud computing trouve ces origines et sa philosophie dans l'histoire de l'informatique elle-même; les services offerts par cette technologie sont basés sur des machines très puissantes appelées Datacenter (qui ont le même concept philosophique que les ordinateurs centraux), ces Datacenter sont composés de serveurs (qui ont le même concept philosophique

que les mini-ordinateurs) et qui offrent des services de logiciels et d'infrastructures, ces services sont accessibles via des navigateurs ou des applications web (qui ont le même concept philosophique que l'architecture client/serveurs et les protocoles Internet); d'un autre côté, les limites en capacités de stockage et traitement imposés par les appareils mobiles a poussé les chercheurs a associé ces derniers a des machines (Datacenter) capables d'héberger et de stocker les applications et les données dont les utilisateurs d'Internet ont besoin, et par la suite offrir des applications Cloud accessibles via un ordinateur personnel ou par l'intermédiaire d'un appareil mobile.

D'un autre coté l'évolution des services Internet a suivi un itinéraire logique et draconien qui a conduit au développement des services Cloud. Tout à débuter en 1980 avec des entreprises de fourniture d'accès aux réseaux de recherche régionaux aux Etats Unis d'Amérique tel que PSINet, UUNET et Netcom. Et c'est seulement en 1989 que le premier fournisseur d'Accès Internet (FAI) via le réseau téléphonique, est devenu opérationnel. La première génération des fournisseurs d'accès Internet FAI 1.0 ne permettent d'offrir qu'un accès limité au réseau Internet via le réseau téléphonique, mais avec la propagation rapide et gigantesque de ce réseau, les fournisseurs de services web ont étendu et amélioré les services offerts par Internet ce qui a conduit à la deuxième génération des FAI 2.0 qui permit aux navigateurs web d'utiliser la messagerie électronique et d'accéder aux serveurs mail. La troisième génération, le FAI 3.0 a connue l'apparition des centres de colocations qui permet d'offrir, au navigateur web, la possibilité de télé-communiquer et de se connecter avec d'autre réseaux Internet, d'accéder aux serveurs de stockage et traitement, de sécuriser les ressources physiques et le transfert des données. L'investissement des fournisseurs d'accès Internet sur la consolidation des infrastructures réseaux, a rendu possible la création et la fourniture de services d'applications sur Internet ce qui a conduit à la quatrième génération des FAI 4.0. Un fournisseur de service d'applications (FSA) (appelé aussi fournisseur d'applications en ligne) est une entreprise qui fournit des logiciels ou des services informatiques à ses clients a travers un réseau (Internet en général) [16]. Le plus grand intérêt de ce modèle est de fournir un accès à des applications particulières (comme un programme de facturation médicale) en utilisant un protocole standard comme le protocole http [17]

En 1999 Google et salesforce.com ont lancé le premier site web public pour les services Cloud. Ces services représentent l'extension logique et nécessaire des quatre générations FAI, L'apport principal des fournisseurs de services Cloud ou FAI 5.0 est qu'ils permettent à un groupe d'utilisateur de partager les mêmes ressources physiques (sous les model Infrastructure-as-a-Service ou Storage-as-a-Service par exemple) et logique (Software-as-a-Service ou Desktop-as-a-Service par exemple) en même temps et à moindre cout. Les FAI 5.0 fournissent également un modèle économique, qui permet aux utilisateurs, de payer seulement les ressources qu'ils utilisent associées à un contrat de niveau de services (Service Level Management (SLA) en anglais), qui permet aux fournisseurs de services Cloud de déterminer et garantir un certain niveau de qualité à leurs clients.

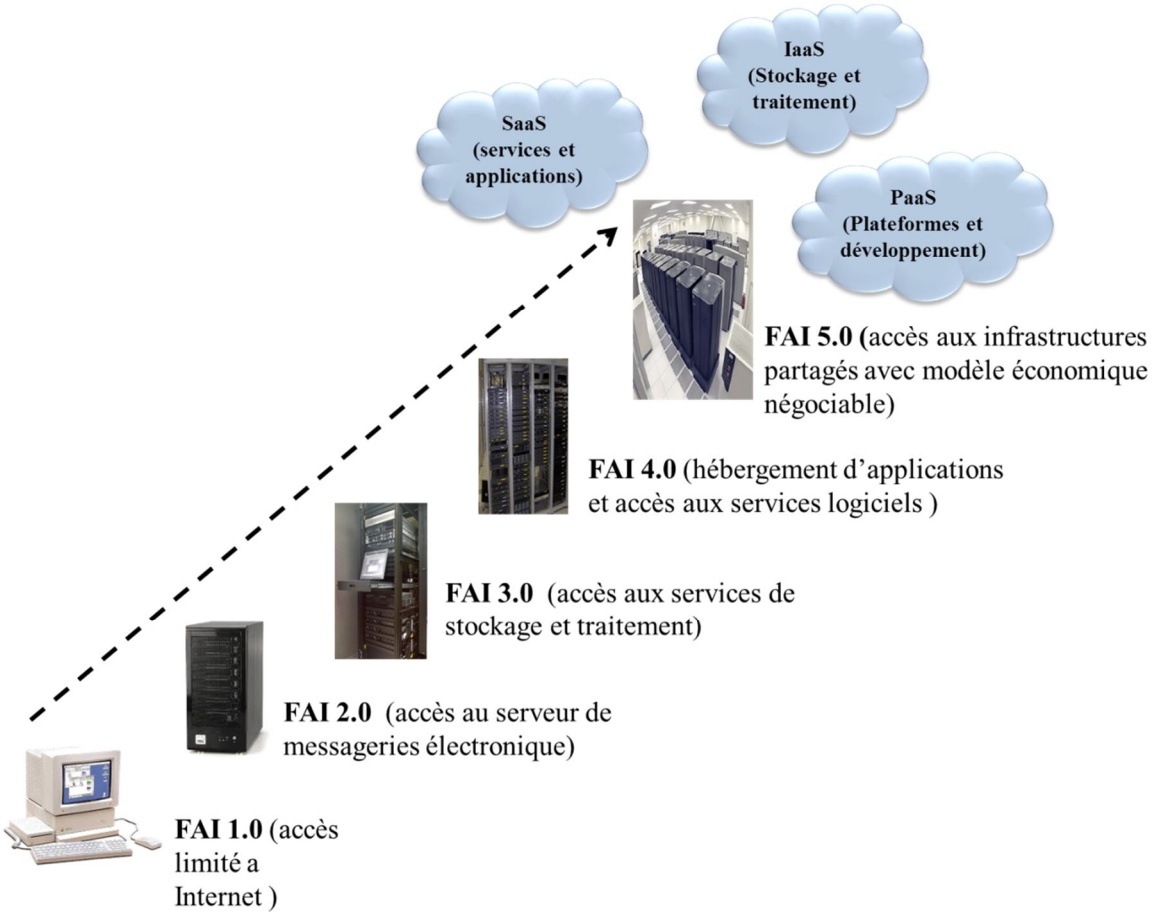

Figure II.1: Evolution vers le Cloud.

II.2. Principe du Cloud computing

L'ambiguïté qui entoure le concept du Cloud computing rend sa définition très difficile. Généralement un système de calcul (computing) puissant offre à ces utilisateurs un ensemble de ressources physiques et logiques gérées éventuellement par des outils de virtualisation et de coordination afin d'assurer un maximum d'efficacité.

L'efficacité des systèmes Cloud computing proviennent essentiellement de leurs dynamismes. Ce caractère permet aux manageurs des systèmes Cloud computing de gérer de façon dynamique et intelligente [18] les différentes ressources et applications offertes par le Cloud. Un Cloud manageur (que ce soit humain ou simple programme) peut facilement et rapidement de décider d'activer, de désactiver ou bien supprimer une ressource de l'ensemble du système, selon son jugement sur l'apport de cette ressource par rapport au reste du système [19].

Le fait que les ressources d'un système Cloud computing sont généralement distribuées [20] sur des zones géographiques distantes impose l'utilisation d'un logiciel centralise qui permet de dénombrer les ressources physiques et virtuels disponibles, ainsi que les tâches à exécuter pour chaque utilisateur, afin de distribuer ces derniers de la façon la plus optimale possible, selon le contrat de niveau de service de chaque client.

Un client Cloud computing peut allouer, augmenter ou réduire le nombre et la capacité des ressources qu'il utilisent au cours du temps, selon ces besoin en espace de stockage ou micro-processeur, ce qui permet aussi au manageur Cloud computing de calculer avec précision la capacité des ressources utilisées par chaque client, et par la suite, établir la facture de chaque client, sur la base de ces besoins. Autrement dit, le modèle économique du Cloud computing permet aux clients de dépenser leurs argents selon leurs besoins.

Quand un client a besoin de plus de ressources (Ex. Espace de stockage, logiciel, machine virtuel) il n'a qu'à exprimer sa demande auprès du fournisseur du service Cloud, et c'est ce dernier qui doit auto-approvisionner ces ressources pour le client. Autrement dit le client n'a pas à ce soucier de l'exécution de ses tâches mais seulement juste de bien les définir.

Dans la littérature, on présente généralement les systèmes Cloud comme une extension nécessaire et logique du Grid computing, sauf qu'il ne s'agit pas seulement de cela, mais aussi d'un système de virtualisation de ressources qui permet aux manageurs de taches de partager les mêmes ressources physique pour plusieurs clients en même temps, ce qui permet un accès simultané de milliers d'utilisateurs et une mise en ligne efficace et optimale de ces derniers sur Internet.

La virtualisation des ressources physiques permet, d'un côté, l'accès de plusieurs clients à la même ressource physique, ce qui optimise l'exploitation de cette dernière; et d'un autre côté, de réduire les couts pour le client, qui au lieu d'allouer tout un serveur, n'aura qu'à allouer une machine virtuelle, avec les capacités dont il a besoin [21].

Afin de couvrir les besoin des clients en matière de ressources physiques, virtuelles et logiques, les concepteurs des systèmes Cloud ont déterminé trois modèles de base (qu'on verra avec plus de détails dans la suite de ce chapitre):

1. Software-as-a-Service: ce modèle permet d'offrir les services d'un logiciel qui peut être partager entre plusieurs clients [22]. On peut dériver de ce modèle plusieurs autres modèles comme Security-as-a-Service, Desktop-as-a-Service … etc.
2. Platform-as-a-Service: ce modèle est destiné aux développeurs d'applications Cloud computing, il permet d'offrir un ensemble d'outils de développement logiciels en ligne.
3. Infrastructure-as-a-Service: ce modèle n'est généralement pas accessible par des clients Cloud ordinaire; vue qu'il permet de gérer l'ensemble des ressources physiques et virtuelles du système Cloud. On peut dériver de ce model trois principaux modèles : Network-as-a-Service, Storage-as-as-Service et Computationnel-as-a-Service.

II.3. Définition du Cloud computing

Afin de discerner la plus grande partie de la notion Cloud computing nous allons présenter un ensemble de définitions qui éclaircissent chacune une facette du Cloud:

Selon [23] le Cloud computing est un type de calcul qui offre un accès simple et sur demande, d'un ensemble de ressources informatiques hautement élastique. Ces ressources sont fournies comme service sur Internet, et sont maintenues grâce à des séries d'innovation et

d'entretiens. Le Cloud permet aux utilisateurs de minimiser le cout d'utilisation, en se délaissant de se préoccuper de la façon dont les ressources son gérer ni où elles se trouvent.

Selon [24] le Cloud est un vaste regroupement de ressources virtuelles, facilement accessibles et utilisables. Ces ressources peuvent être dynamiquement reconfigurées pour s'adapter à une charge variable, ce qui permet une utilisation optimale des ressources. Ce regroupement de ressources est généralement exploité par un modèle pay-per-use, dans lequel des garanties sont offertes par le fournisseur de services au moyen des contrats de niveau de services.

Selon [25] le Cloud computing est un modèle qui permet d'accéder au réseau de façon ubiquitaire, facile et sur demande. Ce modèle offre la possibilité de partager et configurer les ressources Cloud (réseaux, serveurs, stockage, applications et services) qui peuvent être rapidement approvisionnées et libérées avec un effort de gestion et des interactions minimales.

Selon [26] le Cloud computing est un modèle émergeant, de développement, déploiement et livraison, qui permet d'offrir des services en temps réel sur Internet.

Selon [27] le Cloud computing est un modèle de service qui combine entre: le principe général de fourniture des technologies de l'information (IT),composants infrastructurels, approche architecturale et modèle économique – fondamentalement, une confluence de Grid computing, virtualisation, service à la demande, hébergement et Software-as-a-Service (SaaS).

En analysant ces cinq définitions on constate que la définition du Cloud tourne au tour de ses principales caractéristiques, qu'on peut diviser en deux parties:

1. Du point de vue fournisseur de services Cloud: avec les définitions [23, 24] et [25] on trouve les caractéristiques suivantes: l'élasticité, mis à jour automatique, évolutivité massive, auto-approvisionnement de ressources, et ressources partagées.
2. Du point de vue utilisateur final du Cloud: avec les définitions [26] [27], on trouve les caractéristiques suivantes: l'instantanéité, la disponibilité, accès sur demande, pay-asyou-go (On vera ces caractéristiques en détails plus loin dans ce chapitre).

Le Cloud computing n'est qu'un ensemble d'outils de virtualisation et de management de ressources, basé sur de puissantes machines. Ces machines sont regroupées en un seul système, dont les ressources sont éventuellement distribuées, et qui doit assurer et satisfaire un ensemble de caractéristiques, et cela pour faciliter l'accès à un large spectre de ressources.

II.4. Caractéristiques du Cloud computing

Dans cette section nous vérrons les principaux caractéristiques du Cloud computing.

II.4.1. Auto-guérison

Tout système Cloud doit contenir une ou plusieurs copies de chaque application déployée en lui, de telle façon qu'en cas de disfonctionnement de l'application en cours, l'application en copie vient la remplacer en prenant l'état actuel de l'application en échec. Les

applications en copies doivent être maintenues et mise à jours à chaque fois que l'application en cours est modifiée [28].

II.4.2. Multi location (Multi-Tenancy)

Sur le Cloud une même application peut être utilisée par plusieurs clients en même temps, en préservant la sécurité et les données privées de chaque client. Cela est possible en utilisant des outils de virtualisation qui permettent de partager un serveur sur plusieurs utilisateurs [29].

II.4.3. Evolutivité linéaire

Un système basé Cloud computing technologie a la faculté de découper les principales taches du système en un ensemble de petits morceaux, et de les distribuer sur l'infrastructure virtuelle du Cloud [30].

II.4.4. Service orienté

Tout système Cloud est basé sur un ensemble de services indépendants les uns des autres. Ce qui donne de la puissance à la technologie Cloud c'est le rassemblement de ces services en une seule unité afin de présenter un service Cloud qui couvre tous les niveaux sur lesquels une application est basée (Niveau virtuel, niveau logiciel, une interface facile à utiliser …. Etc.) [31]

II.4.5. SLA (Service Level Management)

Avec les services Cloud, un client peut négocier le niveau de service qu'il lui convient et il doit payer pour cela. Dans le cas où les ressources Cloud sont en surcharge, le système crée d'autres entités d'applications Cloud en utilisant les outils de virtualisation disponible afin de respecter les termes du contrat SLA [32].

II.4.6. Virtualisation

Un système basé Cloud computing est un système complètement virtuel et indépendant de la couche physique sur laquelle les ressources et applications sont déployées [33].

II.4.7. Flexibilité

Les services Cloud sont destinés à supporter les lourdes comme les petites charges de travail en augmentant ou réduisant automatiquement les ressources utilisées selon la tailles des taches a traiter [34].

II.5. Principaux modèles de livraisons des services Cloud

Dans cette section nous vérrons les principaux modèles de livraison des services Cloud.

II.5.1. Le modèle Software-as-a-Service (SaaS)

Avant l'apparition des technologies et modèles Cloud computing, les clients achètent des logiciels en mode téléchargement ou livraison à domicile, ce qui met le client mal à l'aise par rapport aux cinq principaux éléments:

1. La mise à jour nécessaire et continuelle du logiciel.
2. Le respect de la licence d'utilisation du logiciel.
3. Le payement brusque d'une somme importante, ainsi que le payement pour chaque mise à jour.
4. Les performances de l'ordinateur du client, en termes de capacité de stockage et d'exécution du logiciel acheté, ainsi que la compatibilité du système d'exploitation.
5. La sécurisation de l'échange de données, notamment pour les logiciels opérants sur des réseaux publics.

L'avènement du modèle de déploiement Software-as-a-Service, a complétement bouleverser l'industrie logiciel dans le monde, avec ce model le client alloue un logiciel installer, stocker et exécuter sur le Cloud [35]. Par rapport au modèle précèdent, le modèle Software-as-a-Service représente des facilités considérables pour la navigation et l'exploitation des ressources physiques et logiques sur le Net [35]:

1. La mise à jour du logiciel est automatique, et c'est l'entreprise Cloud qui s'occupe de cela.
2. La licence de l'utilisation d'un logiciel sur le Cloud est toujours associée à un contrat de niveau de service, qui assure au client un certain degré de qualité de service et clarifie les droits et devoirs du client vis-à-vis de l'entreprise qui offre le logiciel sur le Cloud.
3. Le Client ne paye qu'un abonnement annuel ou mensuel ou bien il ne paye que ce qu'il a utilisé, selon le système de payement du fournisseur de service Cloud.
4. Puisque le logiciel alloué par le client est exécuté sur le Cloud, ce dernier n'a pas à se soucier de l'infrastructure physique (Compatibilité avec le côté matériel) ni le côté logique (compatibilité avec le système d'exploitation).
5. Enfin, les clients payent généralement une charge supplémentaire pour que le fournisseur de service Cloud s'occupe de la sécurisation des données de ces clients.

En plus de ce qu'importe le modèle Software-as-a-Service de nouveautés, ce dernier représente des avantages directs pour l'entreprise de fourniture ou de consommation des service Cloud:

1. La mise en place des infrastructures, serveur d'hébergement, système d'exploitation, et personnel de gestion est entièrement assurée par les fournisseurs des services Cloud, ce qui permet aux consommateurs SaaS d'externaliser les ressources dont ils ont besoin et par la suite réduire les couts et gagner du temps.
2. Les fournisseurs des services Cloud peuvent facilement contrôler l'accès et la mise à jours des logiciels qu'ils fournissent, ce qui leurs permet de réduire l'utilisation frauduleuses de leurs produits
3. L'établissement d'un contrat de vente et d'utilisation de logiciel entre le fournisseur et le consommateur, permet d'un côté, d'établir un plan de revenu économique stable

pour l'entreprise qui fournit le SaaS, et d'un autre côté, établir un certain niveau de confiance que le fournisseur acquiert envers ces clients avec le temps.

4. Avec les outils de virtualisation, qui deviennent de plus en plus puissant, les fournisseurs des services Cloud fournissent leurs services selon le modèle one-to-many, ce qui permet de partager un logiciel (avec l'infrastructure qui le supporte) à plusieurs clients en même temps. Cela réduit les coûts de déploiement du SaaS et le personnel requis pour la gestion des ressources Cloud.
5. Le consommateur est complétement loin de tout détail concernant le matériel ou le système d'exploitation avec lequel il accède au SaaS, puisque ce dernier est accessible via n'importe quel navigateur web.
6. Le consommateur peut facilement configurer son logiciel via une API (Application Programming Interface) que le fournisseur met en place pour ces clients pour qu'ils puissent configurer le SaaS selon leurs besoins.

II.5.2. Le modèle Platform-as-a-Service (PaaS)

Ce modèle représente une variante du modèle SaaS, sauf qu'au lieu d'offrir une application comme service il offre un environnement de développement d'applications comme service, ce modèle permet au développeur d'utiliser et réutiliser un large spectre d'outils, de composant logiciels, et de blocs de codes ce qui lui permet de développer rapidement son application. Avec ce modèle les fournisseurs des services Cloud offrent un ensemble d'outils et d'environnement de: développement, exécution, déploiement, distribution et de payement des applications crées au niveau de la plateforme. Le fournisseur de service d'une Plateforme de développement est aussi destiné à produire et continuellement proposer et développer des standards ainsi que des outils de développement pour faciliter la tâche au programmeur et développeurs d'applications Cloud. Une Plateforme de développement sur le Cloud permet de réduire considérablement les couts de déploiement des applications avec une grande rapidité de propagation vu que les fournisseurs des services Cloud placent assez de canaux de communications et de liaisons entre l'application développée et l'infrastructure de déploiement [36].

Avec le modèle PaaS les développeurs peuvent créer et déployer des applications: web mobile, Cloud ou bureau sans avoir besoin d'installer un outil dans leurs ordinateurs de travail. D'un autre coté l'interopérabilité des Platform-as-a-Service permet aux développeurs de créer et de déployer leur application à partir et sur n'importe quel système d'exploitation sans avoir à se soucier de la gestion des compatibilités entre les composants de l'application et ceux du système hébergeant [37].

La fourniture de serveur d'hébergement ainsi que d'outils de modélisation et de programmation, à moindre cout, permet de dispenser les entreprises ainsi que les boites de développement d'achats de ressources matérielles et logicielles (serveurs, outils de virtualisation et d'administration … etc.) nécessaires au développement de leurs applications, ce qui rend les PaaS, les modèles les plus attractives et les plus convenables pour le développement et le déploiement d'applications dans le web.

L'un des principaux buts des Platform-as-a-Services est la démocratisation du web, ce qui veut dire, rendre le web accessible sur tous les niveaux: consultation, évaluation, contribution, et développement d'applications. Si les trois premiers niveaux sont accessibles pour tout le monde avec les fournisseurs de services classiques (on dit services classique par rapport aux services Cloud); le dernier niveau n'est accessible que par des développeurs web

spécialisés dans le domaine, notamment pour la création et l'administration des sites web, et la maitrise d'outils de développement tel que: Java, J2EE, JavaScript/Dojo… etc. Avec le modèle PaaS le développement d'applications web dévient plus facile, puisque il dispense les développeurs non expérimenté dans le domaine, d'apprendre l'utilisation des environnements et les outils telle que Eclipse, Netbeans, et Microsoft access, pour enfin déployer ces applications manuellement sur les Infrastructure-as-a-Service.

Un service PaaS est caractérisé par un ensemble de points qui le différencie des Plateformes de développement sur un bureau classique:

1. Un outil de développement offert comme service sur une Plateforme Cloud est accessible par plusieurs développeurs en même temps; au contraire des environnements de développement tel que Microsoft Visual studio qui ne sont destinés qu'à un seul utilisateur; ce qui permet aux développeurs de communiquer efficacement.
2. Maintenir l'évolutivité de l'application développée est l'une des tâches les plus importantes des Plateformes Cloud, sauf qu'avec une Plateforme classique la gestion de l'évolutivité ne se fait que lors du déploiement de l'application, mais avec le model PaaS l'évolutivité est gérée en temps réel au cours du développement de l'application. Cela permet aux développeurs de bien ajuster ces paramètres afin d'adapter les composants de leurs applications aux Infrastructures de déploiement et d'avoir une exécution et un niveau de service optimal.
3. Le modèle PaaS offre aux développeurs la capacité de contrôler et de surveiller l'exécution de leurs applications après le déploiement de ces dernières, au contraire de la totalité des environnements de développement classique qui n'offrent pas aux développeurs les outils nécessaires afin de suivre leurs applications après les avoir déployer sur le web.
4. A l'encontre des environnements de développement intégré tel que Microsoft Visual Studio qu'on doit payer (et payer chaque nouvel version téléchargée), et ensuite installé sur sa propre machine. Un outil de développement sur une Plateforme Cloud est utilisé directement sur le Cloud avec un abonnement ou une facture basée sur ce que le développeur a à utiliser comme outils et moyen de déploiement.

Prise en charge	**Plateforme traditionnelle**	**Plateforme Cloud (PaaS)**
Point d'accès	La plupart des API ou appareil (bureaux, mobile) sont pris en considération.	Utilisation de navigateurs web.
Interaction avec les fournisseurs d'outils de développements.	Plusieurs fournisseurs sont pris en charge.	Définis par la plateforme Cloud.
Framework de développement .	Intégration de plusieurs Framework (ex. J2EE, .NET…etc.)	Définis par la plateforme Cloud.
Serveurs d'applications	Plusieurs fournisseurs sont pris en charge.	Fournis par la Plateforme Cloud.
Base de données	Plusieurs fournisseurs sont pris en charge.	Fournis par la Plateforme Cloud.
Outils de Virtualisation	Plusieurs fournisseurs sont pris en charge.	Fournis par la Plateforme Cloud.

Capacité de stockage	Plusieurs fournisseurs sont pris en charge.	Fournis par la Plateforme Cloud.

Tableau II.1: Comparaison entre Plateforme de développement traditionnel et Plateforme dans le Cloud [16]

Ce qui fait la force des Plateformes Cloud computing est qu'elles sont principalement composées d'un studio d'outils de développement accessible via un navigateur web tel que Mozilla Firefox ou Internet explorer, ces outils doivent impérativement avoir des liens qui leurs permettent de ce relier aux différents services web et bases de données externes. Le modèle PaaS offre également des outils de surveillance et de contrôle d'applications qui permettent aux développeurs de suivre et comprendre l'évolution de leurs applications et d'apporter, au fil du temps, des améliorations. Les aspects les plus importants pour le développement d'une application Cloud sur une Platform-as-a-service, son accompagnés par un outil d'assistance automatique qui permet de garantir le développement d'une application massivement évolutive, fiable et avec un grand niveau de sécurité, sans avoir à configurer ou à payer quoi que ce se soit de plus (par rapport aux couts d'utilisation de la Platform Cloud). Ces outils d'assistance automatique doivent répondre au besoin du développeur, tout au long du cycle de développement (Création d'interfaces, création et administration de base de données, codification et développement, tests, documentation, déploiement, statistiques … etc.). Ces outils doivent maintenir à la fois la sécurité et la propriété intellectuelle du code final de ce modèle, ce dernier doit représenter un atout et une opportunité considérable pour le développement d'applications sur le web, et doit avoir un système de payement basé sur ce que les développeurs utilisent comme outils de développement.

II.5.3. Le modèle Infrastructure-as-a-Service (IaaS)

Il est nécessaire de remarquer que les fournisseurs d'infrastructures classiques et celles basées sur le Cloud permettent d'offrir des ressources matérielles et logicielles qui permettent d'héberger et d'offrir des services sur le net. La différence cruciale entre ces deux modèles est qu'un fournisseur IaaS sur le Cloud met en œuvre un système de payement basé sur le taux de consommation des ressources physiques et virtuelles, qui peuvent évoluer selon la demande du consommateur.

Avec le modèle IaaS le fournisseur peut contrôler l'augmentation, la réduction et l'accès aux ressources de l'infrastructure. Ce contrôle facile et efficaces des capacité de l'infrastructure Cloud permet au fournisseur d'héberger des applications web, ou bien, d'aller plus loin en hébergeant des software-as-a-service, des Platform-as-a-service ou des outils de support pour l'amélioration des applications hébergées, ce qui permet aux entreprises d'externaliser leurs ressources et de se débarrasser de la gestion du personnel et des ressources matérielles et logicielles nécessaires à l'hébergement de leurs applications [38].

Les services offerts par le modèle IaaS ressemble à ceux offert par le modèle SaaS, sauf qu'au lieu de payer les services d'un logiciel en ligne on paye pour l'obtention de capacités de traitement de microprocesseur, les capacités de stockage de disque dure, les capacités d'affichage de cartes graphiques …etc. Avec le modèle IaaS le consommateur paye pour qu'il soit libre de tout engagement ou préoccupation par rapport à l'évolutivité, la gestion, la sécurisation, la virtualisation, et le partitionnement des ressources et données. Du point de vue client le modèle IaaS permet au consommateur de réduire la distance entre les ressources

Cloud d'un côté et entre lui et ces derniers d'un autre côté, et cela en choisissant pour chaque serveur l'application à exécuter. Du point de vue fournisseur, le model IaaS lui permet d'attirer tout client désirant un système qui lui permet de déployer, gérer, et avoir une évolutivité massive de ces applications, en ne payant que ce que le client consomme.

En résumé on peut dire qu'une Infrastructure-as-a-Service doit regrouper trois principales caractéristiques:

1. Evolutivité de l'infrastructure selon les besoin des clients; et qui doit être presque en temps réel.
2. Le client achète quand il veut, et la quantité de ressources dont il a besoin, et il ne paye que ce qu'il utilise.
3. L'accès aux meilleures technologies de pointe (best-of-breed technologies), pour le stockage, traitement et virtualisation des ressources.

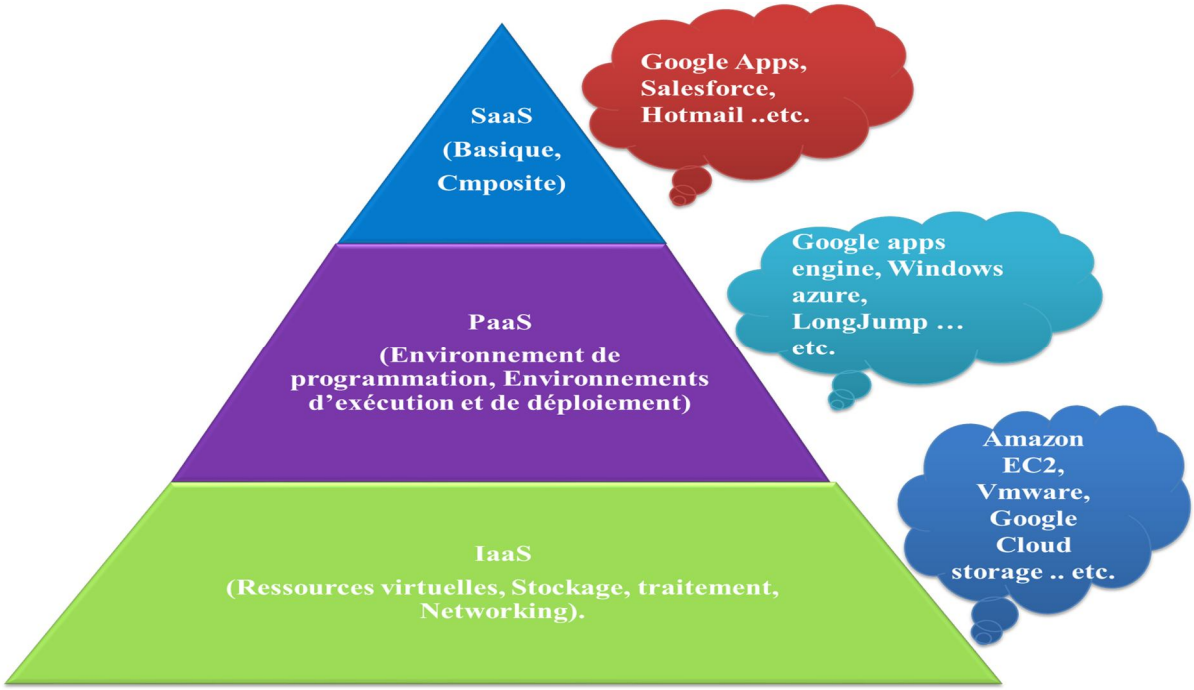

Figure II.2: Principaux modèles de livraison Cloud.

II.6. Modèles de déploiement du Cloud

En plus des modèles de livraison qui permettent de concrétiser les services Cloud, on trouve un ensemble de modèles de déploiement de services basés Cloud computing. Ces modèles permettent de définir le degré d'accès de l'utilisateur final aux fournisseurs de services Cloud. Ces modèles sont divisés en quatre grandes catégories.

II.6.1. Cloud Publique

Le Cloud publique, appelé aussi Cloud externe [39], représente le Cloud traditionnel utilisé par la majorité des clients sur Internet. Avec un Cloud public les ressources sont auto-approvisionnées dynamiquement via des applications ou des services web, possédés par des

fournisseurs de ressources Cloud qui partagent ces derniers, et qui produisent des factures pour cela.

La Supervision, la sécurisation et l'hébergement des ressources Cloud est gérée par une partie tierce, qui devrait offrir ces services à tout client désirant utiliser des ressources Cloud. Ces ressources sont généralement installées sur un ou plusieurs Datacenter afin qu'un plus grand nombre de clients possibles puissent partager les mêmes ressources en même temps.

Le contrôle de l'accès et la sécurisation des ressources matériels et logiciels est contrôlé entièrement par la partie tierces qui possèdent les ressources Cloud, ce qui limite la liberté des clients par rapports au contrôle et à la configuration de leurs application et ressources Cloud.

II.6.2. Cloud Privé

Le Cloud privé, appelé aussi Cloud interne est ce genre de Cloud qu'on trouve sur des réseaux privés et qui n'est accessible que par un ensemble de clients qui ont l'autorisation d'y accéder [40]. Principalement ce modèle permet d'avoir plus de contrôle sur la sécurité des données et applications sur le Cloud, facilite la gestion des ressources Cloud par le propriétaire de ces derniers, et augmente la fiabilité du système Cloud. Mais d'un autre côté, les organisations qui ciblent ce genre de modèle, doivent eux-mêmes posséder, gérer, et réduire le personnel nécessaire à la construction et utilisation des ressources Cloud, ce qui nécessite de grands investissements d'argent et de personnel pour mettre en œuvre leur Cloud privé.

La différence principale entre un Cloud privé et un Cloud publique est le fait que les ressources d'un Cloud privé sont destinées seulement au clients autorisés par l'organisation qui possède les ressources, et ça ne peut pas être partagé avec d'autres clients de l'extérieur. On distingue trois types de Cloud privés:

1. Cloud privé global: c'est un genre de Cloud privé où les applications et ressources physiques et virtuelles sont regroupées sur une infrastructure possédée par l'organisation, et gérée par des experts internes.
2. Cloud privé collectif (en communauté): c'est un genre de Cloud privé où les ressources sont possédées par une partie tierce, mais allouées, gérées et utilisées par une organisation unique qui détermine une SLA lui permettant d'avoir le contrôle total sur les ressources.
3. Cloud privé géré: c'est un genre de Cloud privé ou les ressources sont possédés par l'organisation consommateur et géré par un fournisseur Cloud.

II.6.3. Cloud Hybride

Avec le modèle hybride, les organisations peuvent associer le Cloud privé avec le publique, de cette façon, ces organisations peuvent déployer leurs principales et sensibles applications sur leur Cloud privé et déployé le reste sur le Cloud publique, ce qui peut réduire considérablement les coûts de construction et de management des ressources sur le Cloud.

II.6.4. Cloud Collective (en communauté)

Il s'agit d'un genre de Cloud où les ressources sont partagées par plusieurs organisations. Cette communauté d'organisation peut partager les tâches de gestion de ces ressources, comme la sécurisation des données, le déploiement d'applications, l'authentification … etc. ces ressources, qui peuvent être installé dans ou hors-sites (les sites des organisations de la communauté), peuvent être gérées par une partie tierce comme par une des organisations faisant partie de la communauté [41].

II.7. Cloud computing: bénéfices d'utilisation

Plusieurs types d'utilisateurs peuvent bénéficier, de façon directe ou indirecte, des services offerts par le Cloud [16].

II.7.1. Utilisateur particulier

De nos jours les clients sur Internet, utilisent le Cloud (une grande partie, sans le savoir) pour le stockage: d'email, d'information concernant la vie privée, des photos personnelles, de la musique … etc., en utilisant également des services Cloud, pour achat et vente de différents types de marchandises, ou pour l'échange et la communication d'informations sur des réseaux sociaux telle que Facebook, LinkedIn, MySpace … etc. Le Cloud sert également comme moyen d'hébergement de site et d'applications web développées par des utilisateurs non expérimentées. Avec le Cloud, un simple navigateur Internet peut organiser des réunions, des téléconférences, des parties de jeux vidéo ou même de simples appels téléphoniques. Tout ça avec des possibilités de stockage et une rapidité de traitement très évolutive. Et avec un niveau de services prédéterminer, entre le client et les fournisseurs des services Cloud.

Malgré le fait que les fournisseurs des services Cloud garantissent la protection des données privées des utilisateurs particuliers, le risque d'une attaque sur les systèmes Cloud ou la destruction accidentelle des informations est toujours un risque très probable même dans le Cloud.

II.7.2. Entreprise particulière

Ce qui rend le Cloud particulièrement attirant pour les entreprises particulières c'est les coûts de développement très bas pour la création et l'hébergement d'applications ou de sites web sur le Cloud, en sachant que l'utilisation des produit de ces entreprise, ne nécessitent presque aucun payement, ce qui augmente le nombre de clients.

Une entreprise particulière peut bénéficier directement des services de publicités pour mettre en ligne des annonces de ces produit, ou d'utiliser des services de vente telle que ebay, ou même avoir des contrats de financement avec des banque électronique qui se trouve sur le Cloud.

II.7.3. Société au début d'évolution

L'enjeu principal pour les entreprises qui débutent dans le domaine économique est essentiellement la création d'une plateforme très développée technologiquement, qui permet la commercialisation rapide et efficace des produits de cette entreprise. La solution la plus

rapide et la plus sûre dans ce cas-là est l'externalisation des ressources et des plateformes, afin de réduire les couts d'installation et de gestion du matériel nécessaire, cet opération peut être garantie et à moindre cout avec des fournisseurs de services Cloud. Autrement dit l'enjeu de ce genre d'entreprise est d'avoir un ensemble d'outils et de canaux qui leur permet d'avoir une rapidité d'action pour adapter leurs produits au changement rapide du marché. Pour cela les sociétés qui sont encore à leurs début, doivent adopter une solution qui leurs permet d'avoir une certaine flexibilité quant à la gestion de la portabilité et l'interopérabilité des plateformes d'hébergement de leurs produits.

II.7.4. Petites et moyennes entreprises

Beaucoup de contraintes doivent être prise en considération lors de l'évaluation des PME (Petit et Moyenne Entreprises) à savoir: Nombre de produit, chaine de productivité, âge de l'entreprise, fond d'action, et les lois internes et externes qui régissent le fonctionnement de ce genre d'entreprise … etc. La sécurisation de données privées, ainsi que la collaboration efficaces entre les unités de production, est un facteur très important dans la croissance et l'enrichissement des PME. D'un côté, le personnel informatique de ce genre d'entreprise est très limité en nombre et en compétence, ce qui réduit l'efficacité de l'équipe IT (Information technologies) responsable de la collaboration interne et externe des PME et la publicité des produits. Avec l'externalisation de la gestion des applications qu'offre le Cloud, de telle entreprise peuvent évoluer plus rapidement et plus surement sur un large spectre de marchés, en garantissant une mise-à-jours durable et une utilisation continue des technologies de pointe.

II.7.5. Grande entreprise d'affaire

Avec la croissance exponentielle des données et des applications utilisées et contrôlées par les grandes entreprises (telle que les entreprises Internationales ou la société multinationale), l'utilisation de mécanismes et de systèmes hautement évolutifs devient une nécessité incontournable pour ce genre de société. Le Cloud computing représente l'alternative la plus adéquate pour stocker et héberger tout genre de données et d'application pour ce genre d'entreprise à moindre cout. Avec le Cloud les grandes entreprises d'affaires peuvent utiliser des moteurs de recherches spécifiques aux données et services dont ils ont besoins, comme ils peuvent utiliser des applications dans le Cloud telle que des applications de gestion: de document, d'achat et de vente, et d'accéder également à une large base de clients afin d'avoir leurs avis sur les produit et le niveau de service qu'offre ces entreprises.

Malgré le fait que la sécurisation des données et des applications utilisées ou déployées sur le Cloud par ces entreprises, représentent un enjeux très important, la réduction des couts ainsi que le gain important sur le temps qu'une entreprise peut perdre (dans l'achat et le maintien des ressources et le recrutement des personnel nécessaire à la gestion des applications), rend le Cloud l'unique possibilité pour une évolution rapide et efficaces des Grande entreprises. D'un autre côté, l'enjeu de ce genre d'entreprise est de proposer des architectures informatiques assez souples et flexibles pour permettre d'avoir une publicité attractive et une rapide propagation sur l'échelle mondiale, cela nécessite l'utilisation de technologie de pointe basée sur le Cloud afin de permettre une mise à jours automatique et continue, pour arriver à ces fins.

II.8. Cloud computing: avantages et inconvénients

Le Cloud computing est comme toutes technologies, il a des avantages comme il a des inconvénients [42]:

II.8.1. Avantages

- Réduction des couts de gestion: avec le Cloud les entreprises non plus à se soucier de la gestion des ressources ou du personnel nécessaire à la supervision de leurs plateformes… il n'ont qu'à former le corps principal du personnel à utiliser les applications Cloud visés par l'entreprise.
- Environnement amical: la virtualisation permet d'économiser l'énergie utilisée pour l'alimentation des ressources, et réduit l'émission du CO_2.
- Système anti désastre: la récupération des données et des applications après un désastre (séisme par exemple) est gérée par un backend qui stocke et relance le système à nouveau pour assurer une disponibilité permanente des services Cloud [43].
- Mobilité et accès facile: en stockant nos données et en déployant nos applications sur le Cloud l'accès à ces derniers devient seulement une question de connexion Internet.
- Réduction des couts d'utilisation: le modèle économique du Cloud permet aux clients de réduire et de Controller leurs dépenses, puisque ils ne payent que ce qu'ils utilisent comme ressources Cloud.

II.8.2. Inconvénients

- Conformité réglementaire: lors du transfert des données du client vers le fournisseur du service Cloud, le client est seul responsable de l'intégrité et de la sécurité des données.
- Dépendance des services: un client n'a pas la possibilité de changer le type de services à consommer chez un fournisseur donné.
- Vie privée et lois de confidentialités: les lois qui régissent la préservation des données privées est différente d'un pays à un autre (par exemple les USA n'a pas les mêmes lois que L'union européenne), ce qui peut mettre les clients en situation de confusion par rapport aux services Cloud à utiliser, et cela peut limiter le transfert de données d'une ressource Cloud vers une autre qui se trouve dans un autre pays.
- Récupération de données: le fait de segmenter les taches et les données stockées par les clients et ensuite les éparpillés sur l'infrastructure Cloud, rend leur récupération et par la suite leurs rassemblement très difficile.
- Identification des clients: avec l'utilisation croissante du Cloud et l'utilisation multi-location de ces ressources, il devient de plus en plus difficile d'identifier par qui et de quel endroit les données ont était modifiées?!
- Stockage de données: le stockage physique des données dans le Cloud est effectué par les fournisseurs des services ce qui limite la manipulation de ces dernier par les clients.

II.9. Contrôle et supervision des données dans le Cloud

L'un des sujets les plus vifs et le plus importants liés à la technologie Cloud est la gestion des données. Le cycle de traitement des données passe par plusieurs étapes: construction, modification, sécurisation, stockage (élimination), et contrôle [44].

II.9.1. La nature des données

Le nombre de plus en plus croissant des clients utilisant des services Cloud ainsi que leurs diversités, a engendrés l'apparition de différents types de données qui nécessitent chacune, un traitement spécifique. Puisque une donnée image ne peut pas être modifiée comme un simple texte, un contrat de vente ne peut pas avoir le même niveau de sécurité qu'un email professionnel et les données d'un Blog ne peuvent pas être stockées comme un ensemble de données structurées… etc.

Les types de données ne sont pas le seul enjeu, mais également la quantité de données à gérer sur Cloud. Pour pouvoir stocker et contrôler l'accès à des millions de vidéos ou d'images sur des réseaux sociaux, ou pour gérer de gigantesques masses de données structurées et générer chaque seconde par les entreprises multinationales, il faut avoir des capacités de stockage énorme et des applications efficace et disponible 24h/24.

Avec une telle quantité et une diversité importante des données, les entreprise ainsi que les particuliers ont besoin d'accéder rapidement avec un minimum de latence, afin qu'il puise opérer avec un maximum de données pour augmente leur compétitivité dans un marché incertain.

Les systèmes Cloud doivent également assurer un bon niveau de collaboration entre leurs ressources afin de permettre un accès intégral et un contrôle totale des données par les différents intervenants dans la chaine de productivité jusqu'au consommateur.

II.9.2. La sécurisation des données

Les trois principaux axes sur lesquelles se base la sécurisation des données dans le Cloud sont:

- L'emplacement des données.
- La Manipulation des données.
- Le transfert des données.

II.9.2.1. L'emplacement des données

Le problème le plus important à soulever dans ce cas-là est les lois qui régissent l'accès et le contrôle des données dans chaque pays, puisqu'une fois les données envoyées dans le Cloud, le client ne peut plus les contrôler, puisque seul le fournisseur des services Cloud peut décider de l'emplacement géographique dans lequel les données seront stockés. Le risque réside dans le fait que les lois d'accès aux données diffèrent d'un pays à un autre, ce qui peut

donner l'autorisation (selon les lois de certains pays), d'accéder aux données des clients, pour des raisons de sécurité nationale par exemple.

Un autre problème est la possibilité de stocker les données des clients particuliers avec les données des entreprises ce qui les expose aux dangers d'une attaque de pirates ou de virus.

Les données doivent également être protégé par des SLA qui empêche des sociétés de publicités de les utilisées pour des fins commerciales.

Enfin les Métadonnées qui servent comme moyen de description de données sont déterminées au préalable par le fournisseur ce qui réduit le contrôle des données par le client.

II.9.2.2. Manipulation des données

Un fournisseur de service Cloud digne de confiance doit avoir un contrôle total sur les données de ces clients en mettant en œuvre un ensemble de mécanismes qui vise à : vérifier l'exactitude et l'intégrité des données stocker et traitées, manipuler le type de fichier qui sert à stocker les données, contrôler la cohérence entre les données en entrée et les données en sortie, crypter les données sensibles et restreindre l'accès aux personnes qui sont autorisés par l'SLA, avoir des mécanismes de récupération de données en cas de failles et protéger le niveaux physique de tout accès non autorisé, et enfin s'assurer de la supervision des données (sous demande du client) de tout emplacement de stockage (même les données redondantes).

II.9.2.3. Transfer de données

Il est essentiel de suivre quelques conseils de sécurité afin d'avoir un transfert sûr de données. Avant toute opération le client doit s'assurer que l'espace de stockage du fournisseur Cloud ne contient aucune faille qui peut conduire à une fuite de données, ensuite le client doit choisir une piste sure entre lui et le fournisseur pour que personne n'intercepte ces données.

Afin de garantir un transfert sure de données le fournisseur de services Cloud utilise un Réseau Virtuel Privé (RVP), qui contient un par feu qui sépare l'Internet et les ressources Cloud et un système de cryptage afin de coder les données pour que seul l'ordinateur ou la ressource de destination (qui possède la clé de décryptage) puisse les lires.

II.9.3. Traitement de données

Un fournisseur de services Cloud doit s'assurer que son système trouve toujours les ressources dont il a besoin quand il a besoin afin de pouvoir stocker et traiter la masse d'information de plus en plus croissante que produisent les entreprises multinationales à travers le monde, cette masse peut être divisée en trois catégories:

1. Les données à grande échelle: le Cloud offre un cadre important pour le traitement de ce genre de données (ex. la recherche scientifique en génomique computationnelle, le suivi et l'analyse des fréquences radio-étiquetés d'identification, l'analyse des flux des nouvelles en temps réel … etc.), en se basant sur un ensemble de techniques telle que: MaPreduce de Google et Hadoop de Apache, afin de créer des systèmes massivement évolutives qui permettent l'auto-approvisionnement d'espace de stockage et d'unités de traitements selon les besoins des clients.
2. Base de données dans le Cloud: avec la complexité de plus en plus croissante des applications dans le Cloud et la nature distribuée des données à traiter, le modèle

relationnel rencontre beaucoup de difficultés pour répondre aux requêtes complexes. Une solution serait de répliquer les données sur chaque serveur ou bien les partager sur plusieurs applications, mais ce genre de solutions pourrait compliquer la fourniture des données à la demande. Afin de résoudre ce problème, les grandes entreprises de fourniture de services Cloud ont créé leurs propre base de données Cloud: Google Bigtable qui ressemble à une grande table pour le stockage et le partage de données basée sur Mapreduce pour l'interrogation et le traitement de données, Amazone simpleDB qui représente un services web d'indexation et de stockage de données, Cloud-based SQL qui est une BD relationnelle dans le Cloud basée sur la Platform Microsoft Azure… etc. il y a des tentatives aussi de créer des bases de données Cloud open source, comme: MangoDB pour le stockage de document coder en C++, CouchDB de l'organisation Apache … etc.

3. Les données d'archivage**:** l'archivage de données est un domaine très ancien en informatique qui consiste à stocker des données statiques et rarement utilisé (les règlements légaux (selon chaque pays) obligent les entreprises de les stocker). Il existe plusieurs modèles d'archivage dans le Cloud tel que : IBM SmartCloud™ Content Management (http://www-935.ibm.com/services/us/en/it-services/smartcloud-archive.html), Microsoft Exchange Online Archiving(http://office.microsoft.com/en-us/exchange/microsoft-exchange-online-archiving-archiving-email-FX103763589.aspx) … etc. qui ont pour but d'externaliser la gestion et le stockage de ce genre d'informations rarement utilisées.

II.9.4. Métadonnées

Les Métadonnées représentent le moyen le plus sûr pour comprendre et mapper les données des clients afin d'éviter un quelconque mélange entre ces derniers et les données d'autres clients. Les Métadonnées servent également de moyen de distinction des données lors du transport de ces derniers d'une ressource Cloud vers une autre pour s'assurer qu'ils seront véhiculés vers la bonne destination.

II.10. Manipulation du Cloud

Les fournisseurs de services Cloud offres un ensemble d'outils qui permettent la manipulation et la configuration des applications et des ressources dans le Cloud selon les besoins de chaque client [44]:

II.10.1. Plateformes de développement

Dans cette section nous présentons quelques platformes de dévelopement de systèmes Cloud computing.

II.10.1.1. Les Framework web

Un Framework ou cadre en français offre aux développeurs web un ensemble de librairies qui leurs permettent de réutiliser les codes dont ils ont besoin sans avoir à les réinventés à chaque fois, afin de créer des sites, des applications ou des services web. On compte plusieurs Framework qui on été adoptés par des entreprises offrant des services de Plateforme de développement comme:

- Zend : qui est un Framework open source basé sur PHP 5.3 offert par l'entreprise Jelastic Cloud Housting.
- AJAX: est un Framework très célèbre basé sur JavaScript et XML et qui permet de développer des applications et des sites web dynamiques afin qu'ils peuvent extraire les données qu'il traite de manière asynchrone à partir des serveurs de destination. Entre autres, AJAX a était intégré comme Framework de développement web avec Microsoft Windows Azure Cloud Platform vue les avantages qu'ils offrent en terme de communication et d'interaction avec les serveurs.
- Django: c'est aussi un Framework open source basé sur python qui peut être combiné avec une base de données SQL Google Cloud, afin de développer des applications sur le web.
- … etc.

II.10.1.2. Hébergement d'applications

Les services d'hébergements web représentent les applications les plus importantes dans le monde du Cloud, et c'est d'ailleurs le sens même de « Fournisseur de Service Cloud ». Par exemple, MOSSO est une maison d'hébergement Cloud fondée par des employés de RACKSPACE qui offrent des services de stockages pour des applications d'entreprises, l'interface MOSSO est basée sur une multiplateforme en cluster, qui permet aux développeurs ainsi qu'aux concepteurs logiciels d'exécuter des applications Windows, linux, Mac, … etc. en se servant d'un système de stockage de fichiers illimités basé sur un réseau de livraison « Limelight Networks », outre, les concepteurs on a la possibilité de distribuer leurs fichiers et applications selon l'architecture qui leur convient sur des centaines de serveurs afin d'assurer une disponibilité totale de leur services (http://www.rackspace.com/cloud/).

II.10.2. Utilisation d'applications sur le Cloud

Les fournisseurs de services Cloud tel que Google ou AmazoneEC2 offrent une variété d'applications SaaS que les clients peuvent utiliser en ligne. Par exemple en suivant ce lien www.google.com/enterprise/apps/business/ on peut accéder à tous les SaaS qu'offrent Google pour ces clients. Mais si le client ne trouve pas l'application qu'il cherche, soit il effectue une recherche en utilisant un moteur de découverte et composition de services Cloud (SaaS, PaaS ou IaaS) ou bien il peut développer sa propre application en utilisant des fournisseurs Platform-as-a-Service tel que www.salesforce.com/platform/overview/.

II.10.3.Navigateur web

Le moyen le plus répandu pour accéder à des services Cloud c'est les navigateurs web. Pour cela les plus grandes entreprises tel que Microsoft (IE8) ou Mac (Sfari 3.1) ont adaptés et améliorés leur navigateur web afin qu'ils prennent en considération les derniers standards et technologie web.

D'autres navigateur web orienté Cloud computing sont également apparues sur le marché comme: **Maxthon Cloud (**fr.maxthon.com/**),** Puffin 3.0 (www.puffinbrowser.com), PocketCloud Web (www.pocketcloud.com/), Cloud Web Browser (cloudwebbrowser.sourceforge.net/) …etc. Ces navigateurs permettent d'augmenter la capacité de Transfert de données vers le Cloud, améliorer la portabilité et l'interopérabilité

multiplateforme, synchronisation du stockage et du téléchargement de données sur le Cloud, facilitent l'affichage, l'accès et la recherche de fichier à distance … etc.

II.11. Adoption de la technologie Cloud

La technologie Cloud a était adoptée par les plus grandes entreprise à travers le monde, vu les avantages qu'elle offre **:**

II.11.1. Google Apps Engine

Avec Google Apps Engine l'entreprise Google offre un ensemble d'outils qui permettent à ses clients de facilement s'adapter et développer des applications et ensuite les déployer rapidement sur le Cloud. Google Apps Engine s'occupe de la configuration des machines virtuelles, de l'évolutivité des ressources de stockage et de calcul, ce qui permet aux développeurs de se concentrer seulement sur la saisie et l'optimisation du code source. Google Apps Engine offre également un ensemble d'APIs, bibliothèques et composants réutilisables comme l'authentification et les services de mailing ce qui permet aux développeurs de réduire le temps de développement de leurs applications. Enfin, le déploiement d'application à grande échelle est l'une des principales taches de Google Apps Engine, grâce à la réplication automatique de données et l'équilibrage de charges, ce dernier permet aux applications web de garder un bon niveau de qualité de service [45].

II.11.2. Microsoft Windows Azure

Windows Azure est une Plateforme de développement basée sur un système d'exploitation Cloud qui permet aux développeurs de créer leurs applications sur la base d'un ensemble standard tel que REST, SOAP ou HTTP. Windows Azure permet également aux développeurs de configurer dynamiquement les serveurs virtuels ainsi que l'espace de stockage selon les besoins de chaque application [46].

Windows azure offre également un ensemble de services pour:

- La gestion de base de données dans le Cloud, avec SQL-based web service, ce qui permet, d'accéder à ces derniers à distance, par d'autres partenaires, ou par des utilisateurs mobiles.
- La réutilisation de composants, avec les services .Net, qui offrent un ensemble d'outils pour accélérer le développement et le déploiement d'application dans le Cloud.
- La fourniture d'un ensemble de kits, avec services Live, qui représente un centre de documentation, d'API et d'échantillonnage pour la mise en route des applications basées Windows azure.

II.11.3. Services Web Amazon

Amazon est l'un des leaders les plus importants de l'industrie Cloud dans le monde. Avec l'ensemble des services Cloud qu'il offre ainsi que les contributions que proposent Amazon, ce dernier est devenu le fournisseur de service Cloud le plus connu dans le monde. Parmi les services offerts par Amazon, on trouve: Amazon ElasticCompute Cloud (Amazon EC2) pour le redimensionnement et la configuration dynamique des ressources de calcul, Amazon SimpleDB pour le stockage et l'indexation automatique de données dans le Cloud,

Amazon Simple Storage Service (Amazon S3) pour la fourniture d'interface web simple pour la synchronisation des traitements des données stockées par Amazon SimpleDB, Amazon CloudFront pour la livraison et la diffusion distribuées des données à travers le globe en utilisant des point et des sites d'approvisionnement éparpillés un peu partout sur Internet, Amazon Simple Queue Service (Amazon SQS) pour la sauvegarde et le stockage de messages lors de l'acheminement vers leurs destination, afin d'éviter la perte de données, et de faciliter la visualisation de message par tous les intervenant dans la chaine de communication sur le Cloud (clients, développeurs, partenaire d'affaire …etc.) [47].

II.11.4. IBM

IBM offrent essentiellement un service de consultation et d'aide à l'immigration vers le Cloud, qui permet un suivit complet des clients lors du transfert de leur données et applications vers le Cloud. IBM propose un modèle économique sur lequel se base l'évaluation des couts d'immigration vers le Cloud, par la suite IBM fournit des conseils métiers qui permettent aux clients de prendre les bonnes décisions quant à la création et l'utilisation de leurs Cloud privé/public. Sur le plan sécurité, IBM propose une architecture unifiée qui vise à ré-architecturer les systèmes traditionnels afin de les adapter aux besoins des clients, notamment pour l'isolement l'authentification, l'accès et la gestion de ressources virtuel dans le Cloud [48].

II.12. Pourquoi le Cloud?

Cette question peut être posé d'une autre façon plus claire : pourquoi le Cloud computing au lieu du Grid computing ? Le Cloud computing est un terme général d'abstraction qui encapsule l'ensemble des processus et opérations sous-jacentes qui s'exécutent sur des infrastructures et qui sont cachées aux yeux des utilisateurs finaux. Ces utilisateurs ne peuvent voir et gérer leurs plateformes ou applications qu'à partir d'APIs, contribuant ainsi à créer un niveau d'abstraction qui leurs permet d'interagir avec des plateformes très puissantes de façon rapide et facile, et cela bien que les opérations infrastructurelles réelles soit plus complexes et nécessitent des ressources virtuelles et physique que l'utilisateur ne peut pas gérer tout seul. A ce stade-là, le Grid et le Cloud sont similaires, sauf qu'avec le Grid le client bénéficie d'un ensemble de ressources, reparties géographiquement et déployées sur des Grilles de calcul et de stockage [49], mais se basant rarement sur les technologies de virtualisation, et avec un modèle d'accès restreint à un public bien ciblé, ce qui rend difficile l'utilisation du Grid computing pour offrir des services au large publique sur Internet. A l'encontre du Grid, le Cloud est entièrement basé sur des technologies de virtualisation, ce qui permet d'exploiter la couche physique de façon optimale et facilite l'exposition et l'offre de services au large publique. Les services basés Cloud ne sont pas destiner seulement à la fourniture de ressources pour le stockage, calcul ou networking, mais également pour fournir des services de haut niveau tel que les applications (Ex. SaaS ou PaaS) avec un modèle de payement basé sur le taux de consommation de ressources.

II.13. Cloud computing: Un modèle pour le future

Les prédictions futures des plus grandes industries spécialisés en TI (Technologie de l'Information) révèle l'importance et l'espace que cette technologie va occuper dans le futur. Par exemple : L'IDC a évalué le revenu des entreprises Cloud pour l'année 2008 a 16.2

miliaire de dollars, et qui a augmenté de presque 2 milliaires en 2009, et qui est arrivé jusqu'a 26.8 milliaire en 2013 ,ce qui est équivalent à 26% TCAC de croissance chaque année (Taux de Croissance Annuel Composé). Gartner a évalué le chiffre d'affaires totales des investissements Cloud en 2008 à 9.1 milliaires de dollars qui est arrivé à 26.6 milliaires en 2013 (soit 24% TCAC). Winter Green Research a évalué le chiffre d'affaire Cloud a 20.3 milliaires de dollars en 2009, et a prédit une croissance qui peut arriver jusqu'a 100.4 milliaires de dollars en 2016 (soit 25,6% TCAC).Markets&Marketsa a évalué le chiffre d'affaire Cloud 37.8 milliaires de dollars en 2010, et à 121.1 milliaires de dollars en 2015 (soit 26,2% TCAC).On remarque que tous les calculs et estimation des taux de croissance indique l'augmentation de la consommation de cette technologie (TCAC entre 24% et 26%) ce qui promet un avenir prospère pour le Cloud.

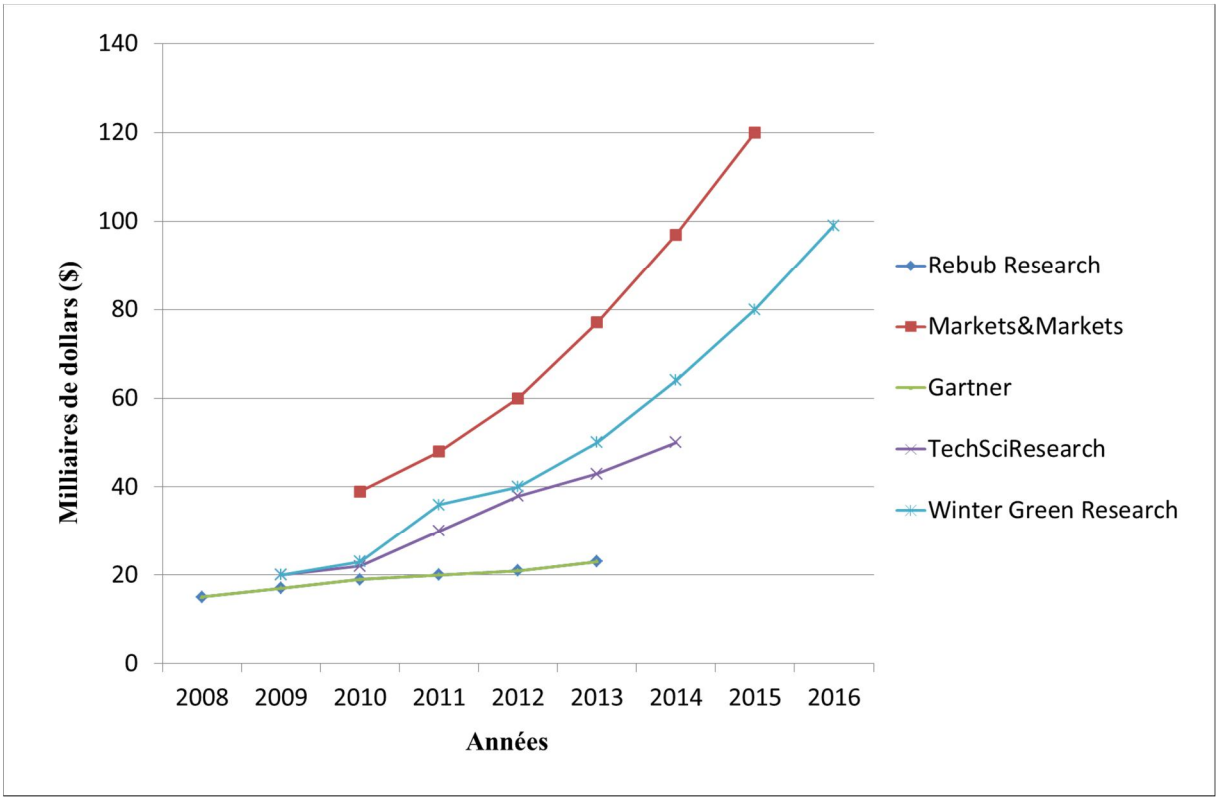

Figure II.3: Prévisions de revenus Cloud Computing [50, 51, 52, 53, 54].

La seule raison qui fait que le Cloud est la technologie d'avenir pour les TI est très certainement les caractéristiques qui le différencient des technologies traditionnelles notamment le système de facturation basé sur les quantités de ressources consommées et la flexibilité de gestion de ressources qui permet aux utilisateurs d'augmenter ou réduire les ressources selon leurs besoins. Cette flexibilité est supportée par un système de gestion automatique d'infrastructure qui permet d'ajuster les ressources selon les besoins des utilisateurs de façon rapide et efficace, ce qui évite la perte de temps.

La technologie Cloud ne cible pas seulement les entreprises qui utilisent des services en ligne, mais également les entreprises qui utilisent des systèmes d'information Interne. De telles entreprises peuvent bénéficier de systèmes Cloud privés accessibles seulement par leurs employés au lieu d'utiliser des applications desktop. Cela permet à ces entreprises de réduire les couts d'achats, maintenance [55] et mise à jours de logiciels (qui ne cesse d'augmenter)

qui sera entièrement pris en charge par les ingénieurs de la société fournisseur du Cloud, et garantie un niveau de compétitivité durable et sure pour tous ces produits. En d'autre terme la responsabilité de la gestion et maintien de ressources est transférée des moyennes et petites entreprises (qui ne peuvent avoir qu'une petite expérience dans ce domaine) vers des grandes entreprises Cloud qui ont des moyens plus importants et des experts plus spécialisés.

Ce qui rend le Cloud computing, la technologie du futur est essentiellement la réduction des couts de déploiement et d'achat de matériels. Les fournisseurs Cloud peuvent réduire leur achat de matériels d'un facteur allant de 5 à 7 en terme de hardware, software, bande passante, consommation d'énergie … etc. [56, 57], et cela en achetant une unique licence pour tous leurs clients. En d'autres termes, un client (que ce soit simple utilisateur ou toute une entreprise) gagne beaucoup plus en déployant et utilisant des services Cloud au lieu d'un service traditionnel.

En résumé on peut dire que l'externalisation (ou sous-traitance) de ressources permet aux entreprises, avec un budget infrastructurelle moyen ou faible, de bénéficier de grande capacités de calcul et stockage, avec un cout minimale et une compétitivité efficace et durable, et cela en libérant ces entreprises des tâches élémentaires et basiques qui concerne la gestion des infrastructures et le maintien des services, pour que ces entreprise se focalisent sur leurs principaux objectifs.

II.14. Les principaux enjeux du Cloud

Dans cette section nous présentons les principales questions et défis rencontrés dans le domaine Cloud:

II.14.1. Problème de Qualité de service

L'une des principales occupations des clients est la qualité des services Cloud [58], notamment la performance et la disponibilité. Afin d'assurer un certain niveau de qualité à leurs clients, les fournisseurs Cloud disposent d'un système de contrôle pour enregistrer les QdS (Qualité de Service) et mis à jour des Machines virtuelles (MV) du système, à chaque fois que la qualité de ces derniers est détériorée. Mais cette solution n'est pas envisageable vu que la migration de données entre MVs peut prendre plusieurs minutes.

Un autre problème est la concentration de plusieurs services Cloud sur le même Datacenter avec une même solution Cloud. Ce qui expose le système à des dangers tels que le piratage, les catastrophes naturelles, ou l'apparition d'une faille système … etc. Et cela peut conduire à la perte des informations des clients. Par exemple, en 2010 une faille dans le système d'hébergement Cloud d'Amazone EC2 a causé le disfonctionnement de 44000 applications hébergées sur ce dernier.

Afin de déterminer un certain niveau de services, les fournisseurs de services Cloud font recours au SLA (Service Level Managment en anglais) qui permet de déterminer la relation entre client et fournisseur, et les responsabilités de chaque partie en cas de détérioration de la qualité de services. Néanmoins, pour le moment il n'existe pas de standard pour l'élaboration des SLA ce qui peut causer des malentendus entre clients et fournisseurs [59].

II.14.2. Problème de transfert de donnés

Le transfert de données intra et inter-Cloud soufre du même problème que celui pour Internet**:** la lenteur. Mais avec le Cloud le problème est plus grave puisque il s'agit de transférer des quantités gigantesques d'informations à partir d'entreprises qui génèrent des dizaines de tira octet chaque jour vers le Cloud. Par exemple avec une bande passante de 18Mb/s, le transfert de 10TO d'un Datacenter situé à Alger vers un Datacenter situé à Helsinki prend environ 54 jours. Amazon par exemple, propose le service AWS Import/Export destiné a transporter les disques durs de stockage aux Datacenter d'Amazon, afin de pourvoir réduire le temps de transfert des données [60].

II.14.3. Problème d'interopérabilité

Une technologie fiable et efficace doit assurer la migration d'applications entre plateformes. L'interopérabilité dans le Cloud est encore un problème très important, due à l'absence de standard qui permet aux fournisseurs Cloud de développer des interfaces et API standardisée [61]. Le problème se pose essentiellement quand il s'agit de transférer une solution Cloud (IaaS ou SaaS) d'un Cloud privé vers un Cloud public exposé au large public. Par conséquent les clients se trouvent devant un choix très limité des infrastructures ou application à utiliser. Outre, la composition de plusieurs services issus de différents fournisseurs tels qu'Engine Yard, Amazone EC2, Google Apps Engine … etc. devient très difficile avec l'absence d'outils de standardisation commune entre ces fournisseurs, ce qui implique de sérieux problèmes de portabilité et d'Interopérabilité. Le tableau ci-dessous illustre les standards Cloud les plus utilisés :

Standards Cloud	**Représentation de données**	**Fonctionnalités**
OCCI (pour Open Cloud Computing Interface en anglais)	XML	(i) Gestion d'IaaS (ii) définition et représentation des capacités de: traitement, networking, et stockage (iii) Interface réseaux et liaison entre sites de stockage [62, 63, 64].
OVF (pour Open Virtualization Format en anglais)	XML	(i) distribution d'application et de plateformes virtuelle (ex. VM et Les SaaS) [65].
TOSCA (pour Topology and Orchestration Specification for Cloud Applications en anglais)	XML et WSDL	(i) description d'applications et d'infrastructures Cloud (ii) représentation des composant et liaisons entre services complexes (iii) facilite la description des comportements des composants Cloud (ex. déploiement, éteindre … etc.) [66].

CIMI (pour Cloud Infrastructure Management Interface en anglais)	XML et JSON	(i) Gestion d'IaaS (ii) contrôler la migration d'applications et données d'un système Cloud a un autre [67].
CDMI (pour Cloud Data Management Interface en anglais)	JSON	(i) Interface d'aide pour le stockage de fichiers (ii) manipulation de données (iii) sécurisation de données (iv) facturation [68, 69].

Tableau II.02: Quelques standards Cloud.

II.14.4. Problèmes de sécurité

La principale question que posent les clients Cloud concerne la transparence de stockage des données. Le problème est essentiellement un problème de confiance. Avec le Cloud les clients ne peuvent pas localiser l'emplacement où leurs données sont stockées, ce qui augmente l'inquiétude des clients vis-à-vis des lois de préservations de confidentialité (selon le pays où se trouvent les données stocké), des catastrophes, des conflits …etc.

Parmi les principales menaces d'ordre sécuritaire sur le Cloud on trouve [58]:

- Confiance entre fournisseur et clients : la relation entre client et fournisseur est essentiellement une relation de confiance, ce dernier doit assurer la confidentialité des données stockées par le client, néanmoins le problème réside dans le fait que les employés d'un fournisseur Cloud peuvent accéder aux données des clients pour les vendre à une société de marketing ou une entreprise concurrente par exemple, ce qui représente une menace réelle [70].
- Clients malveillants: la technologie Cloud est basée sur le partage de ressources entre clients, ce qui veut dire qu'une ressources physique ou virtuelle peut être accéder par plusieurs clients en même temps ce qui augmente le risque qu'un client (pirate ou hacker en anglais) tente d'infiltrer les instances SaaS ou les espaces de stockages d'autre clients pour voler ou inculquer des virus d'espionnages ou de destruction.
- Vérifiabilité des Infrastructures Cloud: puisque les clients (surtouts les fournisseurs de services Cloud de haut niveaux comme SaaS) non pas le droit d'accès ni de contrôle des infrastructures Cloud qu'ils utilisent, ces dernier ne peuvent pas vérifier les versions des logiciels utilisés, la mise à jours des codes, les mécanismes de sécurité … etc. ce qui peut générer des disfonctionnements au niveau de la qualité et utilisation des services Cloud.
- … etc.

II.15. Conclusion

Dans ce chapitre nous avons survolés les principaux concepts et points clés de la technologie Cloud computing, nous avons commencé par présenter le contexte historique de

cette technologie, ensuite nous avons défini le Cloud en exposant ces principales caractéristiques. Nous avons également montré l'impact et l'importance de la technologie Cloud sur les systèmes de fourniture de services actuels, ainsi que les perspectives et les possibilités future qu'offre cette technologie. Enfin, nous avons présenté les principaux défis dont la technologie Cloud computing doit faire face pour améliorer la qualité des services fournis aux clients.

Ainsi dans le prochain chapitre nous commencerons à explorer le sujet de la thèse en présentant les principaux travaux en relation avec l'approche proposée ainsi que leurs catégories.

III
Travaux liés à l'approche proposée

III.1. Introduction

Les services web représentent la technologie la plus exploitée et la plus utilisée pour la fourniture de services, sous forme d'applications, sur Internet. La souplesse et l'adoption de cette technologie par les plus grandes entreprises des technologies d'informations, a conduit également ces derniers à adopter les services web comme Interface aux services Cloud.

La combinaison entre services web et Cloud computing permet d'appliquer les techniques de classifications, découverte et composition des services web sur les services Cloud, mais en prenant en considération les caractéristiques qui différencient une application Cloud d'un service web.

Ce chapitre est organisé comme suit :
Dans la section 2 nous présentons la définition de la découverte de services, les différentes approches de découvertes et les catégories des différents mécanismes et algorithmes de découverte et sélection. Dans la section 3 nous présentons la définition de la composition de service, nous résumons les problèmes liés à la composition et nous présentons les différentes catégories des mécanismes et algorithmes de composition de services web. Dans la section 4, nous présentons les travaux les plus connus et les plus représentatifs de la découverte et composition de services SaaS et Web. Enfin, la section 5 est la conclusion de ce travail.

III.2. Découverte de service web

Dans cette section nous présentons les principaux concepts liés à la découverte de services web.

III.2.1. Définition

La découverte des services web est l'opération qui permet de « localiser une description de service web exploitable et inconnue par les machines sur un réseau, et qui répond à certains critères fonctionnels ou/et non fonctionnels » [71].

Un service de découverte a pour objectif de faciliter la recherche de services sur Internet en se basant sur des agents **:** demandeurs, fournisseurs, ou intermédiaires, qui servent respectivement à formuler la requête, créer les services web et publier ces derniers.

La découverte de service web passe par un ensemble d'étapes:

1. Le service de découverte met le demandeur et le fournisseur en relation.
2. Le service de découverte reçoit les spécifications fonctionnelles et/ou non fonctionnelles que le demandeur cherche.

3. Le service de découverte extrait les descriptions des services web fournies par les fournisseurs (Ex. UDDI, WSDL, Mot clés, Paramètres QoS, OWL-S, DAML-S, RDF, URI … etc.).
4. Le service de découverte utilise un moyen de description sémantique pour mapper et faire rapprocher les spécifications fonctionnelles et non fonctionnelles du demandeur par rapport aux descriptions fournies par les fournisseurs.
5. Le service de découverte exécute un mécanisme de matching pour calculer le degré de correspondance entre la requête du demandeur et les descriptions des fournisseurs.
6. Le service de découverte sélectionne les services web qui correspondent le mieux à la requête du demandeur.
7. Eventuellement, le service de découverte, fournit un service de composition de services web, afin d'avoir des services web composite qui répondent aux différentes parties de la requête du demandeur car il est devenu très difficile de trouver des services web atomiques qui répondent à tous les besoins du demandeur exprimés dans sa requête.

III.2.2. Les approches de découvertes

Les approches de découverte les plus célèbres de nos jours sont :

III.2.2.1. Registre

Un registre est une sorte de pôle centralisé sur laquelle les fournisseurs peuvent publier et placer explicitement, des informations relatives à leurs services [72]. En outre, c'est le fournisseur de service de publication, ou propriétaire du registre, qui a le droit de décider quel type, quand et de quelle manière une information est placée sur le registre. En plus le propriétaire du registre a le pouvoir d'attribuer ou de retirer le droit d'accès aux fournisseurs de services qui veulent placer ou publier leurs services.

Ce pouvoir dont le propriétaire bénéficie, lui impose le devoir de contrôler l'accès des fournisseurs de services au registre, par exemple, le propriétaire de registre doit interdire aux fournisseurs de services de publier des informations sur d'autres services pour des raisons de compétition.

Généralement la technologie la plus utilisé pour mettre en œuvre les approches basées sur les registres est la technologie des UDDI [73]. Mais cette dernière peut également être utilisée pour l'implémentation des Index.

III.2.2.2. Index

A l'encontre des registres, les index son déniés de tout contrôle centralisé ou autorité qui s'approprie le contrôle de la publication d'information. En plus, ceux qui sont destinés à faire des modifications sur les index sont ceux qui les créent, seulement sans l'intervention des fournisseurs des services, qui ne font que fournir des informations sur leurs services.

Avec l'approche Index, toute personne peut créer son propre système de publication de services, et y inclure toutes informations qu'elle désire sur les services qu'elle veut publier et par la suite vérifier et mettre à jours ces informations. Les différences principales entre les registres et les Index ne sont pas seulement dans la centralisation de la publication des

informations relatives aux services web, mais également, dans le fait que la célébrité et l'intégrité d'un Index sur internet ne dépend pas de l'intervention des fournisseurs de services mais du nombre de consultations et le taux d'utilisation de l'index par les puissances du marché.

Généralement les propriétaires des Index utilisent les WSDL [74] pour créer leurs Index, mais la technologie des UDDI peut également être utilisée pour publier des services web.

III.2.2.3. Peer-to-Peer

Avec une approche Peer-to-Peer la découverte des services est entièrement dynamique, ce qui signifie que chaque service web représente un nœud sur le réseau qui a pour fonction de propager la requête du client ver ces voisins, qui doivent par la suite propager à nouveau la requête vers leurs voisins jusqu'à trouver les services web qui répondent à la requête du demandeur [75].

Avec les approches P2P le service de découverte n'a pas besoin de registre de publication de services [76], vu que les services web sont auto-descriptifs et contiennent leur Index de publication et description dans leurs propres plateformes. Ce qui implique que toute information utilisée lors du processus de découverte est à jour, au contraire de l'approche registre ou Index où la mise à jour peut entrainer des temps de latences très importants, ce qui peut réduire la fiabilité des services retournés au demandeur.

L'inconvénient majeur des approches basée P2P est qu'il n y a aucune garantie que la propagation de la requête vers les services web voisins va nécessairement conduire à trouver les meilleur services qui correspondent à la requête du demandeur.

III.2.2.4. Quelle approche choisir?

Préférer une approche à une autre pour l'implémentation d'un service de découverte est loin d'être une question de gout personnel. Chaque approche est destinée à répondre à des contraintes d'environnements bien précis:

- Approche basée registre centralisé: ce type d'approches est destinées aux environnements statiques où les informations sur les services web ne changent pas souvent, vu les difficultés qu'impose la mise à jour de tel registre.
- Approche basée Index : ce type d'approche est plus appropriée aux environnements où le changement d'information et la concurrence entre fournisseur est le moyen qui détermine les critères d'évaluation du marché, vu que dans ce genre d'environnement, c'est la stratégie de commercialisation et de marketing (où les index représentent une partie très importante) qui fait la différence entre fournisseurs.
- Approche basée P2P : il est claire que ce type d'approche est adéquat aux environnements dynamiques mis en œuvre sur des réseaux étroitement liés, vu que la propagation de la requête n'assure pas toujours, l'obtention de résultats efficaces.

III.2.3. Découverte fédérée de services web

Sur un réseau aussi ouvert et colossale comme Internet, la centralisation du contrôle et publication des descriptions de services web est tout simplement impossible, ce qui a conduit à la création de milliers de registres et index sur Internet. La découverte fédérée consiste justement a relié tous ces moyens de publication de services entre eux pour couvrir le maximum possible d'Index et registres afin de les présenter au demandeur, sous forme d'une seule plateforme de publication de services web,et cela pour consolider les résultats retournés à ce dernier [77].

L'idée sur laquelle repose la découverte fédérée est qu'un index ou registre contient des liens vers d'autres index ou registres sur le web, que le service de découverte peut utiliser pour la recherche d'information sur d'autres services web qui répondent à la requête du demandeur. Outre, les Index et registre doivent utiliser une terminologie ou ontologie commune pour éviter les conflits d'interprétation des différentes notions utilisées par les fournisseurs, pour décrire leurs services web.

III.2.4. Catégorisation des mécanismes de découverte de services

Dans cette section nous présentons les différentes catégories de méthodes et approches de découvertes de services.

III.2.4.1. Découverte syntaxique

Les mécanismes de découverte syntaxique sont généralement basés sur la comparaison entre les mots clés de la requête du client avec ceux extraits de la description des services (Ex. WSDL). En utilisant un parseur et un dictionnaire électronique, l'algorithme de matching extrait les mots clés de la requête et de la description du service, et calcule le degré de similarité entre eux en utilisant une représentation vectorielle de chaque partie et en calculant le degré de similarité avec la mesure du cosinus. L'inconvénient majeur des méthodes syntaxique est le non prise en considération de l'aspect sémantique de la requête.

III.2.4.2. Découverte sémantique

La découverte sémantique de services consiste à exploiter les différents concepts qu'on trouve dans une ontologie afin de calculer le degré de correspondance entre les mots clé de la requête et les mots clés de la description. L'implémentation des différents concepts et relations sémantiques, est généralement basée sur deux principaux outils :

III.2.4.2.1. Ontologie

Selon [78], une ontologie est un ensemble structuré de savoirs dans un domaine particulier de la connaissance, ou un ensemble de concepts organisés en graphe dont les relations peuvent être sémantiques ou de composition et d'héritage. Mais pour les services web il n'y a pas une ontologie qui leurs soit spécifique, alors que le développement d'ontologie pour un domaine spécifique (Ex. Médecine, Aéronautique, Finance… etc.) nécessite beaucoup de temps et d'effort.

III.2.4.2.2. WordNet

WordNet est une base de données lexicales pour les termes Anglais [79] (mais il y a d'autres versions pour d'autres langues telles que: WOLF pour le Français)**:** cette base de donnée est composée d'un ensemble de relations conceptuelles sémantiques et lexicale entre les différents mots de vocabulaire anglais, ces relations sont appelées synonymes cognitifs ou Synsets.

En plus de la synonymie, WordNet contient les concepts suivants:

- Hyperonymie: ce sont les termes qui représentent un sens général par rapport à un terme plus précis.
- Hyponymie: ce sont les termes qui représentent un sens spécifique, par rapport à un terme plus général.
- Formes Dérivationnelles connexes: signifie les termes qui sont dérivés lexicalement d'un autre terme et qui partagent un sens commun.

Ces concepts représentent les relations sémantiques entre les mots de la base lexicale WordNet.

WordNet est également considérée comme une ontologie générale, c'est à dire, une ontologie qui ne traite pas un seul domaine spécifique mais tous les domaines qu'on peut décrire avec une langue naturelle. Cela rend WordNet très adéquat au type de moteur de découverte de services qu'on propose dans cette thèse, puisque ce moteur est destiné à la recherche de tout type de services et non pas seulement à des services qui appartiennent à un domaine spécifique. Par rapport aux ontologies, l'utilisation de WordNet peut engendrer des résultats moins précis quand il s'agit de traiter un domaine spécifique, et des résultats plus précis quand il s'agit de traiter divers domaines.

III.2.4.2.3. Découverte dynamique de services

On entend par découverte dynamique la possibilité de localiser automatiquement un service qui répond à des besoins particuliers. Différentes approches dynamiques ont été proposées dans la littérature [80]. Ces approches diverses par le langage de description des services utilisés (ex. DAML-S, logique de description…) et/ou par l'algorithme de découverte utilisé (matchmaking, test de subsumption, réécriture).

III.2.4.3. Découverte non-fonctionnelle de service web

La qualité du service (Quality of Service (QoS) en anglais) représente l'aspect non-fonctionnel des services web. QoS est un ensemble de valeurs qui quantifient ou qualifient les caractéristiques qui différencient un service d'un autre du point de vue qualité des services offerts à l'utilisateur et non pas les fonctionnalités offertes à ce dernier [81].

Voici quelques critères QoS:

- Le coût : est la mesure du coût de demander un service.
- Temps de réponse : le délai de la demande pour obtenir une réponse du service.
- Latence : le délai entre l'arrivée et l'accomplissement d'une demande de service.
- La capacité : est la limite des demandes courantes qu'un service peut traiter.

- La fiabilité : est la capacité d'un service de remplir ses fonctions requises dans des conditions indiquées pendant une période spécifique.
- Débit : le nombre de demandes accomplies pendant un Intervalle de temps.

Avec l'augmentation exponentielle du nombre de services déployés sur Internet et la concurrence illimitée entre fournisseurs de services, qui offrent des services assurant des fonctionnalités presque similaires, il est devenu important d'utiliser la QoS comme critère incontournable pour différencier et choisir (entre les meilleurs services assurant les mêmes fonctionnalités) les services qui assure la meilleure qualité.

III.2.4.4. Découverte de service basée sur l'aspect Infrastructurel

Avec l'avènement du Cloud une nouvelle catégorie de découverte de services est apparue. Les méthodes qui font partie de cette catégorie, implémente des algorithmes de matching pour la découverte et sélection de services basés sur les caractéristiques de l'infrastructure sur laquelle le service est déployé [82]. Ces méthodes traitent des paramètres qui décrivent les capacités des ressources de l'infrastructure cible, telle que : Type d'infrastructure (stockage, calcul, networking), nombre de serveurs disponibles, nombres d'unité CPU, distance géorgique par rapport aux clients … etc. Outre, ces méthodes utilisent de langages spécifiques, telle que RDL (Resource description langage) pour la description des infrastructures. Mais l'inconvénient est l'absence d'un langage standardisé pour la description de ressources et infrastructure Cloud, ce qui peut engendrer des conflits d'interprétation entre fournisseur IaaS, et détériore la qualité de matching entre requête et fournisseur IaaS [82].

III.3. Composition de services

Dans cette section nous présentons les principaux concepts liés à la composition de services web.

III.3.1. Définition

La notion de composition a toujours fait polémique, essentiellement à cause de l'étape de découverte, puisque dans la littérature, une partie de chercheurs considèrent que cette étape est une étape primordiale et faisant partie de la composition, alors que d'autres pensent le contraire :

Selon [83], 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 [84], la composition est un mécanisme qui permet de regroupé 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 est une opération qui permet de créer des chaines de connectivité bien ordonnées à partir de services atomiques, afin de satisfaire des requêtes complexes qui ne peuvent pas être satisfaites à partir de services atomiques isolés.

III.3.2. Requête via Composition

En général un bon mécanisme de composition de service est un mécanisme qui se base sur une étape de découverte qui permet de fournir les meilleurs services atomiques satisfaisants les besoins élémentaires d'une requête complexe. Dans ce cadre on distingue 4 cas où la requête est satisfaite ou non par les services disponibles :

1. Tous les besoins de l'utilisateur (exprimé avec la requête) sont satisfaits par un ou plusieurs services atomiques disponibles. Dans ce cas l'étape de composition peut être négligeable.
2. Seulement une partie des besoins de l'utilisateur est satisfaite par les services disponibles. Néanmoins, les services atomiques composant sont tous approprie a la requête de l'utilisateur. Dans ce cas la base de service doit être élargie et augmenté, et l'opération de composition doit être relancé jusqu'à la satisfaction totale de la requête.
3. Seulement une partie des besoins de l'utilisateur est satisfaits par les services disponibles. Néanmoins, les services atomiques composant ne sont pas appropriés à la requête de l'utilisateur. Dans ce cas le mécanisme de composition doit être optimisé.
4. Tous les services atomiques composant ne satisfont aucune partie de la requête. Dans ce cas, soit la base de services doit être élargie et augmenté, soit le mécanisme de composition doit être optimisé, soit les deux.

III.3.4. Catégorisation des mécanismes de composition

Dans la littérature, nous distinguons deux grands systèmes de catégorisation :

Selon Chakraborty et Joshi [85], les mécanismes de composition peuvent être classés en 4 catégories :

- Composition Proactive : les services sont composés «hors-ligne ».
- Composition Réactive : les services sont composés «enligne».
- 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.

Selon Medjahed [86], les mécanismes de composition peuvent également être classés en 4 catégories :

- Composition Statique : les services sont composés, compilés, et déployés lors de l'élaboration du mécanisme de composition avant même la réception de la requête.
- Composition Dynamique : l'opération de composition de services est lancée lors de la réception de la requête.
- Composition manuelle : les services composites sont créés à la main à l'aide d'un éditeur XML par exemple, sans l'aide d'un moteur de génération automatique de services composites.

- Composition Automatique : l'outil de génération de services composites génère ces derniers sans l'intervention de l'utilisateur qui n'a comme seul tache, la spécification de ces besoins.

III.4. Mécanismes de découverte et composition de services

Dans cette section nous présentons 49 travaux liés de façon directe ou indirecte avec l'approche proposée dans cette thèse. Ces travaux sont classés en 7 principales catégories :

III.4.1. Découverte et composition de services web

Dans l'article [87], les auteurs ont proposé un système qui aide les fournisseurs de services web ainsi que les utilisateurs à améliorer la publication et la découverte des services web. Le système est composé de trois niveaux : (i) Extraction de mots clés : les auteurs ont proposé un mécanisme qui permet d'améliorer, sémantiquement et à l'aide d'ontologies issues de différentes ressources hétérogènes, la liste des mots clés extraites à partir de la requête de l'utilisateur et la description des services web. Les auteurs ont utilisés pour cette fin un algorithme de désambiguïsation. (ii) un processus de matching ontologique : qui permet de trouver les similarités entre les mots clés ontologiques de la requête et la description des services web, pour cette fin les auteurs ont proposé d'utiliser un ensemble de mesures de similarités statistique et sémantiques. (iii) un algorithme de subsomption : qui permet de trouver les descriptions des services web qui correspondent à la requête de l'utilisateur, en utilisant un algorithme qui permet de comparer entre les Input et Output des services web et les mots clés de la requête.

Dans l'article [88], les auteurs ont proposé une approche basée- vecteur d'extraction et WordNet, qui permet, non seulement d'ajouter des informations sémantiques à la représentation vectorielle du modèle d'espace (SVM), mais aussi de réduire la taille et l'espace des vecteurs. La représentation SVM est combinée avec des méthodes basée-noyau (Kernel-based, très répandues et très connues dans le domaine du e-learning) afin de calculer la similarité entre les arbres WSDL pour effectuer les tests de connectivité.

Dans l'article [89], les auteurs ont proposé une approche de découverte et sélection de service basée sur la méthode d'encodage «Prüfer» afin de présenter les services découverts comme une séquence capturant les aspects sémantiques et structuraux des services web. Ensuite les auteurs ont proposé un schéma de matching qui permet de calculer la similarité entre services web. Au début les auteurs ont proposé de modélisé les services web en utilisant une représentation arborescente des fichiers WSDL, par la suite ces arbres sont divisés en deux parties, une partie concrète (information générale sur les services) et une partie abstraite (opération et entrés/sorties) afin de mapper chaque partie avec celle qui lui correspond sur la requête. Ensuite, les auteurs ont proposé deux algorithmes de matching. Le premier permet de comparer entre le niveau abstrait des services, en définissant trois niveaux de matching : nœud de définition, nœud de liaison, et nœud portType. Le deuxième algorithme traite l'aspect abstrait en capturant les informations sémantiques et structurelles des arbres WDL, ce qui permettra de créer une matrice de similarité entre services, ensuite les auteurs proposent l'utilisation de méthode d'encodage « Prüfer » pour la présentation de séquence de services, ce séquence permet de matcher les aspects abstraits entre services à l'aide de la matrice de

similarité. Enfin, le degré de correspondance entre deux services, est calculé par la combinaison entre l'aspect linguistique et l'aspect structurel des services.

Dans l'article [90], les auteurs ont proposé un nouveau modèle de découverte et composition de services Web basé sur les réseaux sociaux (nommé LinkedWS), qui permet la capture des interactions entre services Web. Les relations dans LinkedWS sont: Recommandation/partenariat et Recommandation/Robustesse et collaboration. La composition des services web met en œuvre les relations de recommandation/partenariat et de collaboration, et en cas de panne, elle met en œuvre la relation de recommandation/robustesse. Le maintien de LinkedWS consiste à ajouter des nœuds et des relations ou à les mettre à jour.

Dans l'article [91], les auteurs ont proposé un mécanisme basé IR-Style pour la découverte et composition de services web en introduisant la notion de degré de préférence, qui est basé sur deux paramètres: la pertinence et l'importance des services Web. La pertinence est calculée en utilisant la similarité de cosinus, et le degré d'importance est basé sur le nombre de services web employés dans l'opération de composition. Pour calculer l'importance, les auteurs ont utilisé la distance d'édition entre arbre WSDL, afin de calculer le coût de transformation d'arbres, ensuite les auteurs utilisent une équation itérative pour calculer l'importance de chaque opération d'un service web donné [92]. Enfin, le degré de préférence d'un service web est la somme pondérée de ses scores de pertinences et d'importances.

Dans l'article [93], les auteurs ont proposé une approche sémantique basée sur l'aspect structural et en utilisant une ontologie basée OWL-S pour la découverte de service web. L'idée de l'algorithme est de calculer la similarité entre toutes les entrées/sorties du service web avec ceux exprimés dans la requête en utilisant une ontologie du domaine. Le degré de similarité dépend des relations et inférences ontologique entres les concepts des entrées/sorties. Enfin les auteurs proposent la classification de services web selon le niveau de matching ontologique des entrées/sorties exprimé dans la requête (Exacte, Plug-in, subsume, siblin et échec). En cas d'échec total ou de résultat non satisfaisant, les auteurs proposent l'utilisation d'un algorithme de récupération de services qui peuvent être négligés par le premier algorithme. L'algorithme de récupération est caractérisé par le fait qu'il prenne en considération, tous les aspects de la requête ainsi que du service web (nom de service, commentaires, nom du site web) en utilisant les relations de voisinage structural du domaine ontologique, ce qui permet de récupérer plus de service et d'améliorer les chances d'avoir de meilleurs services.

III.4.2. Découverte et sélection de Software-as-a-Service

Dans les articles [94], [95] et [96], les auteurs ont proposé des Framework basés mécanismes hiérarchiques pour la classification de Software-as-a-Service, afin de sélectionner les meilleurs services. Dans le premier travail les auteurs ont proposé un modèle de classification des attributs Cloud, ce modèle classe les attributs en 3 catégories : positive, négative et modérée, ensuite ils ont proposé un modèle pour calculer le poids de chaque critère ce qui permet de calculer le score de chaque SaaS. Dans le deuxième, les auteurs ont proposé un modèle de classification d'attributs Cloud basés sur des interviews avec des experts du domaine, ce modèle classe les attributs en 6 catégories : Fonctionnalité, Architecture, Ergonomie, vendeur, Réputation, et coût. Ensuite les auteurs ont proposés deux modèles de sélection, le premier sert à comparer entres les paramètres Cloud et le deuxième

entre les produit SaaS de chaque fournisseur Cloud. A la fin les résultats de comparaison sont combinés pour sélectionner le meilleur. Dans le dernier travail le modèle de classification catégorise les attribue Cloud en 8 catégories : Responsabilisation, Agilité, Assurance de service, coût, rendement, sécurité, confidentialité, et convivialité. Ensuite les auteurs ont proposé un modèle de comparaison entre fournisseurs SaaS qui se déroule sur quatre phase : phase de classification et quantification d'attributs, phase de calcul des poids de chaque attribut, phase d'évaluation des performance de chaque service, basé sur les attribut de bas niveaux, et la phase de calcul des scores de chaque service afin de les classer.

Dans les articles [97] et [98], les auteurs ont proposés des méthodes non structurée basée P2P pour la découverte de SaaS. Dans le premier travail les auteurs, ont proposé un schéma de localisation et de recherche de services Cloud en utilisant un protocole de routage sémantique de reconnaissance de topologie, les auteurs ont également utilisé WSDL-S pour décrire sémantiquement les services à découvrir. Dans le deuxième travail les auteurs ont proposé un modèle de publication et découverte de service sur un réseau Cloud P2P, la phase de publication consiste à ce que chaque service enregistre ces informations fonctionnelles et non-fonctionnelle auprès de tous ces nœuds voisins, la phase de découverte consiste à utiliser le modèle probabiliste d'inondation basé sur le trafic sur le réseau.

Dans l'article [99], les auteurs ont proposé un Framework basé attribut dynamique et services web pour la représentation de services et ressources dans le Cloud afin de faciliter la publication, la découverte, et la sélection de services. Au début les auteurs proposent l'extension des fichiers WSDL afin qu'ils prennent en considération les caractéristiques des ressources Cloud. Outre, les auteurs ont présenté la notion de Cluster-as-a-Service qui sert de moyen de regroupement des services et ressources qui se ressemblent, afin de faciliter la publication et la sélection de ces derniers.

Dans l'article [100], les auteurs ont proposé un Framework orienté service pour la réalisation d'un Cloud broker, qui sert d'intermédiaire entre client et service Cloud, ce broker a pour rôle l'exécution d'un système de découverte et sélection de service sur le Cloud basé OWL-S. L'algorithme de matching proposé par les auteurs se base sur deux phases : phase de matching des entrées/sorties : qui consiste à utiliser une approche logique de raisonnement standardisé pour le matching des entrées/sorties des services et la phase de matching des contraintes de chaque services : les contraintes sont soit des contraintes atomiques soit des contraintes de collaboration. Enfin vient la sélection de services Cloud, qui consiste à calculer le score de chaque service à base des degrés de matching des E/S et des contraintes.

Dans l'article [101], les auteurs ont proposé un Framework basé ontologie Cloud pour l'enregistrement et la découverte de service Cloud. Le Framework proposé est une tentative de standardisé la publication de service Cloud, afin d'avoir un système flexible et qui s'adapte efficacement au changement des environnements Cloud.. Les auteurs ont encore proposé une ontologie Cloud qui regroupe un ensemble de concept qui permet de décrire un service Cloud, tel que le type de déploiement, le type de sécurité … etc.

Dans l'article [102], les auteurs ont proposé un système de publication, découverte, et sélection de Software-as-a-Service basé sur une nouvelle ontologie Cloud qui regroupe 650 concepts décrivant les divers aspects d'un SaaS, les auteurs proposent un algorithme de matching qui permet de combiner l'aspect fonctionnel, QoS et Cloud pour la sélection de services. Au début la mesure de cosinus est utilisée pour découvrir les services qui correspondent fonctionnellement aux exigences des clients, ensuite les relations et concepts

ontologique sont utilisées pour combiner les aspects QoS et Cloud afin de mesurer la similarité entre les exigences non-fonctionnelles et Cloud du client, et les attributs de chaque service découvert.

Dans l'article [103], les auteurs ont proposé un comparateur systématique entre fournisseur de service Cloud nommé CloudCmp, pour aider les clients à sélectionner le meilleur d'entre eux. CloudCmp permet de mesurer l'élasticité des ressources, les capacités de stockages ainsi que les capacités réseaux de chaque fournisseur Cloud. Enfin, les auteurs ont proposé un ensemble de mesures pour évaluer chaque aspect infrastructurel fournit par les fournisseurs de services Cloud, comme par exemple : Mise à l'échelle de latence, temps de réponse des opérations, cout par opération … etc.

Dans l'article [104], les auteurs ont proposé un modèle de planification de ressources dans les entreprises de production Software-as-a-Service basé sur la qualité de ces derniers. Ce model QoS prend en considération 6 critères : Fonctionnalité, fiabilité, facilité d'utilisation, efficacité, maintenabilité et affaires. Ensuite, ils utilisent un système de décision multicritères pour sélectionner les meilleurs SaaS. Enfin les auteurs proposent l'utilisation d'un mécanisme de comparaison par paire, afin de pouvoir déterminer les poids de chaque critère QoS.

III.4.3. Composition de Software-as-a-Service

Dans l'article [105], les auteurs ont proposé une approche de composition de service sur le Cloud on utilisant IC Cloud (Imperial College Cloud) comme exemple de démonstration, le but de cette approche est de développer un système qui permet aux fournisseurs de services sur le Cloud de concevoir et composer leurs systèmes de façon rapide et efficace. Au début les auteurs ont défini les composants d'IC Cloud qui consistent en 5 éléments : Machine Virtuelle, entrepôt de services et MVs, services atomique, coordinateur de description, description des exigences. Les auteurs ont également définit trois type d'exigences pour la composition dans le Cloud : besoin en MVs, besoin en ressources réseau, et besoin en disque de stockage. Enfin, les auteurs ont défini le système de composition de services qui se déroule sur quatre étapes: spécification des besoins, sélection des services atomiques, génération et spécification des descriptions en utilisant le langage LCC (pour Language-lightweight Coordination Calculus en anglais), et déploiement sur le Cloud.

Dans l'article [106], les auteurs ont proposé une approche de composition de service Cloud représentée sous forme de services web et basée sur la notion de Skyline (ligne d'horizon). Skyline est exploité pour éliminer les services Cloud qui semblent inadéquats aux exigences du client, et utilise la technique d'optimisation basée-essaims de particules pour sélectionner et composer les meilleurs services. La technique d'optimisation basée-essaims de particules est basée sur le partage d'informations et la coopération entre les particules dans un essaim, ce qui permet de créer un processus de composition de services convergents vers une solution optimale. Les auteurs ont encore présenté un schéma de codage des différents services Cloud ainsi qu'une fonction de fitness basé-variable de décision.

Dans l'article [107], les auteurs ont proposé une approche basée-partage d'informations et modélisation ontologique pour la composition et l'approvisionnement de services Cloud. L'approche proposée est essentiellement basée sur l'exploitation et l'étude des liens sémantiques qui permettent de lier les services complémentaires. Les auteurs ont également utilisé une représentation formelle des liens entre services pour prouver l'efficacité du

mécanisme «Linked Data» qu'il présente pour modéliser les relations entre Infrastructure Cloud afin de créer de meilleurs services composites.

Dans l'article [108], les auteurs ont proposé une approche de recherche et composition de services Cloud basée sur les aspects fonctionnels et non fonctionnels, en exploitant WordNet comme moyen de calcul du degré de similarité sémantique. Les auteurs proposent encore l'utilisation des attributs QoS pour la classification de services. Au début les auteurs propose un Framework de stockage de fichiers WSDL et d'attributs QoS munis d'une table de composition de service, qui contient toutes les combinaisons possibles entre elles. Enfin, cette table est remplie en utilisant un algorithme de matching basé sur la technique classique de Kuhn-Munkres, qui permet de déceler la similarité sémantique entre les concepts véhiculés par chaque opération dans les services atomiques. En plus, les auteurs proposent de modéliser le problème de composition sous forme de graphe orienté et pondéré, où chaque nœud représente un service et chaque bord dénote le degré de correspondance sémantique entre deux services, enfin les attributs QoS sont utilisés pour classer les services composites.

Dans l'article [109], les auteurs ont proposé la notion de service virtuel de composition. Ce genre de service sert à lier deux services exigés par le client et qui ne sont pas composables. Les auteurs ont également présenté un algorithme qui permet d'identifier le service virtuel en cas d'échec de composition, ainsi que ces spécifications. L'idée de l'algorithme consiste à parcourir la chaine de connectivité à partir des inputs du départ jusqu'aux outputs de la fin et le contraire, afin de trouver les informations manquantes, par la suite le service virtuel est créé à partir de ces informations qui manque à chaque service physique. Enfin les auteurs proposent l'utilisation de l'algorithme Graphplan pour l'implémentation de l'algorithme général, choisit pour son efficacité et la richesse d'informations qu'il fournit sur chaque service.

Dans l'article [110], les auteurs ont proposé un model abstrait d'approvisionnement, recherche et composition de services fédérées dans le Cloud, en prenant en compte l'aspect sémantique et QoS de ces services. Au début les auteurs ont défini 3 catégories descriptives pour les services Cloud : description des fonctionnalités, description du domaine, et description de la qualité du service, ensuite, à l'aide d'une ontologie du domaine, les auteurs ont défini un ensemble de règles et définitions qui permettent d'exploiter leur ontologie afin de calculer la similarité sémantique entre input/output et pré-condition/post-condition afin de créer des services fédérées composite dans le Cloud.

Dans l'article [111], les auteurs ont proposé un modèle d'exploitation et de composition de services sécurisés dans le Cloud, le but est de développer une plateforme qui permet de tester, superviser et analyser les services composites par rapport à l'infrastructure et la configuration du système de sécurité. Au début les auteurs ont présenté un système de gestion d'identité des différents services intervenant sur les chaines de composition, ensuite les auteurs ont proposé un modèle en couches, qui prend en charge la création des services composite : couche Cloud pour la gestion de l'aspect infrastructurel, couche plateforme indépendante pour la modélisation des interactions client/service, et couche de modélisation des services composite pour la spécification des exigences des clients.

Dans l'article [112], les auteurs ont proposé une approche de composition de service Cloud basé-attributs QoS. Au début les auteurs proposent l'utilisation d'une méthode décisionnelle pour déterminer si la composition de services qu' il demande (basé sur ces exigences) est faisable ou non, si la réponse est «non», un message est retourné au client pour

qu'il change ces exigences, si la réponse est «oui», les auteurs proposent l'utilisation d'une méthode de composition de services basée sur quatre étape : déterminer les valeurs minimales et maximales de chaque critère QoS, discrétisation des valeurs min et max en plusieurs intervalles qui représentent chacun un niveau de qualité indépendant, pour chaque tache on trouve un ensemble de services candidats en utilisant une méthode d'optimisation de sélection locale, enfin les service sont classés sur la base de leur scores pour choisir le service composite optimal.

Dans l'article [113], les auteurs ont proposé une approche sémantique de composition de service Cloud présentée sous forme de service web, les auteurs proposent également l'utilisation de la décision bayésienne et les chaines de Markov pour analyser les services composites, ainsi qu'un graphe représentant les services découvertes pour illustrer la méthode proposée. Finalement, les auteurs utilisent la mesure du cosinus pour calculer la similarité entre les entrées/sorties des services à composer.

Dans l'article [114], les auteurs ont proposé une approche d'optimisation des attributs QoS basés sur une représentation arborescente des services Cloud. La méthode proposée utilise un algorithme heuristique de recherche pour sélectionner les services à composer. La méthode proposée est basée sur trois principales étapes : création de l'arbre qui représente toutes les chaines de connectivité possible, élimination des itinéraires et chaines illégales et qui semble loin de la solution optimale, et enfin, évaluation et classification des services composites. Les auteurs proposent un algorithme de filtrage qui permet d'éliminer les nœuds de l'arbre de composition (chaque nœud représente un service) sur la base du degré de similarité entre les entrées du service en cours et les sorties du service au niveau qui vient tout juste après, l'algorithme procède également à l'élimination des nœuds en cas où la qualité de service en cours est moins que la qualité demandé par le client. Finalement les auteurs proposent l'utilisation de l'algorithme Best-First Search associé à une fonction de calcul de distance pour chercher les solutions optimales (les meilleurs itinéraires) et classés les services composites.

Dans l'article [115], les auteurs ont défini un modèle de service sous forme de flux, qui a pour but d'être accessible et exploitable par le client de n'importe quel endroit et à n'importe quel moment. Les auteurs ont commencé par définir ce qu'il appelle la fiabilité du service, qui consiste en l'habilité du modèle proposé à composer des services à partir de plusieurs. Ce caractère permet de concevoir un système répondant à un maximum d'exigences des clients et à présenter la combinaison de services comme étant une seule entité. Ensuite les auteurs ont proposé un modèle de combinaison et composition de service basé sur 3 éléments: capteur de contexte, service de contrôle et d'organisation, et service médiateur pour faciliter les interactions avec les clients. Les auteurs ont également pris en considération divers facteurs dans la construction de leurs modèles de composition des services, à savoir : contexte culturel, l'emplacement du client, le temps actuel, comportement humain (triste, heureux … etc.) … etc. Finalement, l'approche de composition proposée est basée sur 5 étapes : captures du contexte et des exigences des clients, identification des services, sélection des services, composition des services et recommandation des meilleurs services.

III.4.4. Architecture basé-Cloud pour la découverte et composition de services

Dans l'article [116], les auteurs ont proposé un modèle Cloud appelé « Service Registryon the Cloud(SRC)», qui représente une extension du modèle basé-mots clés mais déployé comme une application sur le Cloud. Ce modèle vise à découvrir et sélectionner les services les plus appropriés aux exigences fonctionnelles et non fonctionnelles exprimées par le client. Le modèle SRCstock les descripteurs sémantiques des services Web ainsi que l'évaluation de l'état dynamique de la Qualité des services sous forme de fichiers GFS(Google File system) dans le Cloud, utilisent le mécanisme deMap Reduce pour traiter ces fichiers. Outre, l'ontologie utilisée pour calculer la similarité des entrées/sorties entre services est stockée sur chaque instances SRC.

Dans l'article [117], les auteurs ont proposé une approche Cloud computing basée «open standard Extensible Messaging and Presence Protocol (XMPP)» appliquée au domaine de la bioinformatique, qui prend en compte la découverte, l'appel asynchrone, et la définition des types de données pour des services Cloud. Les Services Cloud XMPP peuvent être découverts et être accessible sans la nécessité d'un registre externe. L'appel asynchrone élimine le besoin de solutions ad-hoc, et les entrées/sorties définies en chaque service permettent la génération de clients à la volée sans avoir besoin d'une description sémantique externe.

Dans l'article [118], les auteurs ont proposé une architecture basée Cloud qui implémente une ontologie de découverte pour aider les clients à choisir les meilleurs services basée sur leur QoS. Les auteurs proposent l'utilisation des services web comme interface pour leur service Cloud et une architecture basé sur 8 modules pour lancer le processus de découverte : (1) un portail web pour la spécification des exigences client, (2) service d'administration d'applications pour que le client puisse spécifier l'IaaS sur laquelle il désire déployé son service, (3) service de packaging pour standardisé les informations introduite par le client basé sur « Open Standardization Format » ,(4) service de déploiement d'applications pour découvrir les services adéquates aux requêtes du client basé sur l'ontologie du domaine et les attribut QoS désiré par le client ,(5) service de publication d'IaaS qui permet au fournisseur Cloud de publier leurs produits et services, (6) gestionnaire SLA pour gérer le contrat entre fournisseur et client ,(7) service de contrôle d'SLA pour vérifier le respect des SLA par tous les parties ;(8) fournisseur de service IaaS publié sous forme de services web pour offrir les infrastructure nécessaire aux clients.

Dans l'article [119], les auteurs ont proposé un schéma qui permet d'optimiser l'évolutivité des services dans le Cloud en utilisant la composition comme moyen d'assurer cela. Le schéma proposé se base sur l'analyse des communications entre services ainsi que leur emplacement sur les serveurs Cloud. Au début les auteurs ont défini l'évolutivité d'un service composite comme étant le changement de sa productivité, mesurée à base de sa consommation de bande-passante. Enfin les auteurs ont défini un schéma qui permet de définir l'affectation de chaque service basé sur un algorithme génétique pour avoir un plan de distribution optimale qui maximise la productivité des services composites.

Dans l'article [120], les auteurs ont proposé une approche de composition de services à la demande sur le Cloud en exploitant le caractère de dynamisme chez ces derniers. Les auteurs proposent d'introduire une nouvelle couche intermédiaire appelé Composition-as-a-

Service (CaaS) qui permet de planifier la composition des services, offrir un service de découverte et un service de composition. Enfin, les auteurs ont spécifié les taches dont la couche CaaS est responsable : (1) planification du processus de composition : qui consiste a transformé les description des taches en haut niveau en forme de services web, (2) moteur de recherche : pour chercher les services les plus adéquat au exigences, (3) générateur de processus : pour composer les services précédemment découverts sur la base de leur flux d'entrées/sorties, (4) moteur de raisonnement : qui raffine les résultats de composition en prenant en considération l'aspect contextuel et en appliquant quelques règles définie sur la couche CaaS, (5) Exécuteur de processus : qui produit un environnement d'exécution pour les services composites (6) Contrôleur de processus : qui permet au client de gérer le flux de données de ces services composites.

III.4.5. Architecture basée Agent pour la découverte et composition de services

Dans l'article [121], les auteurs ont proposé une approche basée agent associé à un algorithme de matching basé QoS pour la sélection et le trie de services web, l'architecture est basée sur un agent service web (ASW) responsable de la publication et sélection de service web. Au début l'ASW publie son interface auprès des UDDI sur Internet, afin de facilité au fournisseur de services de le retrouver, ensuite les fournisseurs de services publient les informations et attributs fonctionnels et non-fonctionnels auprès de l'ASW qui procède à la vérification de ces attributs en utilisant son composant de vérification et certification. Quand le client envoie sa requête, un agent de découverte est envoyé pour chercher les meilleurs services qui correspondent aux exigences fonctionnelles et non fonctionnelles, en vérifiant les certificats QoS (produits par l'ASW) publiés sur chaque UDDI par rapport aux paramètres QoS publiés par chaque fournisseur de service. Enfin les services sont classés selon leurs scores QoS en tenant en compte quatre paramètres : temps de réponse, cout, disponibilité, et débit.

Dans les articles [122] et [123], les auteurs ont proposé des approches Grid basée agent mobile pour la découverte et sélection de services. Dans la première approche, les auteurs ont proposé une architecture basée agent mobile pour la gestion des services mobiles dans le Grid. L'architecture est basée sur un modèle hiérarchique d'agents identiques, chaque agent a la capacité de produire des requêtes pour solliciter des services et de migrer d'un terminal à un autre en impliquant l'exécution de processus d'enregistrements et de suppressions. Dans la deuxième approche, les auteurs ont proposé une architecture Grid basée agent mobile pour la découverte et l'accès aux ressources Grid déployées sous forme de services web. le paradigme service web, le protocole SIP (Session Initiation Protocol) et la technologie des UDDI sont utilisés dans ce travail pour permettre au client d'utiliser des agents mobiles pour chercher et accéder aux ressources et applications distribuées sur des terminaux hétérogènes dans le Grid, en utilisant des configurations dynamiques des sessions d'interactions et des fonctionnalités des services déployés sur le Grid en prenant en compte la qualité des interaction et la QoS de ces derniers.

Dans l'article [124], les auteurs ont proposé une architecture basé agents située et mobiles pour la découverte et sélection de services web. Les agents situés ont pour taches : la gestion des ressources du système, l'analyse de la requête pour extraire les mots clés et leurs synonymes, la mise à jours des informations stockées au sein de chaque agent mobile, l'envoi des agents mobiles à chaque réception d'une nouvelle requête, et la réception et l'analyse des

services web retournés par les agents mobiles afin de sélectionner les meilleurs d'entre eux. Enfin les agents mobiles son dispatchés sur Internet pour chercher les meilleurs services web qui correspondent syntaxiquement aux requêtes des clients.

III.4.6. Cloud basée agent pour la découverte et la composition de services

Dans l'article [125], les auteurs ont proposé une architecture basée agent pour la composition de services dans le Cloud. Les auteurs proposent également l'utilisation de services web comme interface pour les services Cloud qui offrent principalement des ressources Cloud, et de modéliser chaque composant du système par un agent spécifique. Le système proposé utilise un service d'ontologie pour la description des besoins client ainsi que les capacités de chaque service Cloud. Le système est composé de quatre type d'agents: (1) agent consommateur : pour formalisé les besoins des client en contactant et en passant des contrats avec les agents intermédiaires et en combinant les résultats obtenus par les agents intermédiaire en un seul services virtuel composite en utilisant un protocole semi-récursive basé agent ,(2) agent intermédiaire (broker) : pour offrir des estimations sur les ressources disponibles selon les besoins de chaque client, (3) agents fournisseurs de services : cet agent est responsable de la gestion d'un ensemble d'agents ressources et de la communication avec d'autre agents fournisseurs de services afin d'assurer la disponibilité des ressources offertes aux clients, (4) agents ressources : chaque agent ressource représente un service Cloud, et il est responsable de répondre aux besoins des clients en ressources, qu'il reçoit de l'agent fournisseur de services.

Dans l'article [126], les auteurs ont proposé une architecture Cloud basée sur des agents mobiles et situés, pour la découverte et sélection de service web. L'approche proposée est basée sur deux régions Cloud, chaque région est gérée par un agent situé responsable de la gestion : des ressources, de l'authentification et de l'accès aux emplacements d'agents mobiles, de l'aspect sécuritaire, et de la création et le dispatching des agents mobiles. La première région est responsable de la réception de la requête du client, en utilisant un agent mobile interface et également effectuée une recherche syntaxique basée mot clé sur les services indexé sur cette région. La deuxième région est responsable de la sélection des meilleurs services qui correspondent sémantiquement à la requête des clients à partir des services (reçus par l'intermédiaire d'un agent mobile de communication) découverts par la premier région, en exécutant un algorithme de sélection basé WordNet.

Dans l'article [127], les auteurs ont proposé un modèle automatique de construction de service composite basé intégration d'agent, Cloud et sémantique. Le principal objectif de l'approche proposée est de développée une méthode qui facilite la réutilisation de logiciel basé service web dans le Cloud. La méthode proposée assure 5 niveau de services : (1) découverte automatique de services web (2) description de services web (3) composition de services web automatique (4) invocation automatique de services web. L'algorithme de découverte et composition de services web est basé sur un double filtre syntaxique et sémantique et utilise un diagramme BPD (Business Process Management Notation) pour modéliser les interactions entre services. Enfin, les auteurs ont présenté l'architecture de leur système composé, de (1) Plateforme Cloud : pour supporter les ressources dont le système exploite et utilise (2) Services web : utilisé par le processus de composition (3) UDDI : qui sert comme registre pour le stockage des information des services web (4) Système multi-agent : pour exécuté l'algorithme de découvertes et composition de services web, cet SMA

est basé sur trois sous-système : sous-système d'analyse pour analyser la sémantique des services web, sous-système de recherche pour la découvertes de services basé sur le double filtre et un sous-système de composition pour déterminé les relation entre services web (5) ontologie : pour la modélisation des connaissance sémantique des services web (6) et enfin des fichier BPEL : pour stocké les services composites.

Dans l'article [128], les auteurs ont introduit un moteur de recherche de service dans le Cloud basé description ontologique. Le moteur proposé met en œuvre trois méthodes pour faciliter la recherche de service dans le Cloud : (1) similarité cognitive : qui permet de calculer la similarité entre deux concepts en comptant le nombre de nœuds voisins, (2) similarité équivalente : qui permet de calculer la similarité entre deux concepts sur la base de leurs valeurs d'étiquetage, (3) similarité numérique : pour calculer la similarité entre deux concepts basé aussi sur leur valeur d'étiquetage. L'architecture proposée par les auteurs est basé sur le trois principaux éléments (1) Agent de découvertes : pour effectue les opérations de: prétraitement de la requête, le calcul des mesures de similarité, et le calcul de la similarité totale basée sur une fonction d'agrégation, (2) Ontologie : qui représente les relations entre services Cloud et regroupant 424 concepts, ce qui a permis aux auteurs de proposer des mesures de similarité sémantique et numérique pour calculer la similarité globale, (3) Interface web : l'interface web permet aux clients d'introduire le nom du service désiré, le type de services cherché (SaaS, PaaS, IaaS, CaaS, et DaaS), et les capacités des ressources Cloud que le client désire avoir.

III.4.7. Architecture Cloud Basée Agent

Afin d'avoir une vue complète sur les travaux et approche Cloud basé agent nous divisons ces derniers en deux catégories :

III.4.7.1. Modélisation des systèmes Cloud basé agent

Dans l'article [129], les auteurs ont exposé un système ouvert de Fédération de Cloud computing basé Agent Mobile (MABOCCF) (Open Cloud Computing Federation Based Mobile Agent en anglais) afin de réaliser un système assurant la portabilité et l'interopérabilité entre les données et ressources dans le Cloud. MABOCCF et composée de plusieurs régions Cloud, Chaque région héberge plusieurs machines virtuelles pour contenir les emplacements des agents mobiles. En outre, chaque région est contrôlée par un ou plusieurs gestionnaires de tâches. Le rôle principal des agents mobiles est de transmettre les données et les codes d'une région à une autre pour assurer la portabilité et l'interopérabilité.

Dans [130] les auteurs ont proposé le concept de Service-as-an-Agent-Service (SaaAS) pour faciliter le déplacement des données, des codes et de la charge du travail, des réseaux Intra-Cloud (réseau LAN) vers des réseaux Inter-Cloud (réseaux WAN comme Internet par exemple) afin d'augmenter l'évolutivité des ressources Cloud et de réduire la consommation de la bande passante. SaaAS est composé de : (1) Agent d'interface : responsable de la coordination entre clients et les autres agents, (2) agent travailleur : pour l'encapsulation des codes et des données afin de les transmettre vers le serveur du domaine ciblé et retourner les résultats de l'exécution aux clients, (3) Agent de gestion du domaine pour gérer les domaines Cloud, (4) Agent de gestion principale : pour contrôler tous les agents de gestions des domaines. En outre, avec SaaAS tous les codes d'exécution et les données sont stockées et dupliquées sur chaque serveur de domaine, afin de rapprocher les codes des données ce qui permet de réduire l'utilisation de la bande passante.

Dans l'article [131], les auteurs ont introduit la combinaison entre la technologie Cloud avec celle du Grid sur une plateforme basée agent mobile. L'architecture proposée par les auteurs, a pour but d'offrir des clusters virtuels avec un contrôle total au client en adoptant spécialement l'aspect sécuritaire des infrastructures Grid, et en profitant du dynamisme des agents mobiles pour la configuration des ressources virtuelles. L'architecture proposée est basée sur trois principaux composants : (1) services : responsables de fourniture des fonctionnalités offertes par la technologie du Grid, (2) Images : qui représente les images des nœuds des clusters virtuels, pour intégrer tous les logiciels nécessaires permettant de gérer le cluster virtuel ainsi que la plateforme d'agent mobile, (3) interface client : pour faciliter l'interaction entre client et le système proposé.

III.4.7.2. Gestion de ressources Cloud à base d'agent

Dans [132] les auteurs ont proposé un model adaptatif basé agent pour l'allocation de Datacenter à base des besoins des clients, la méthode proposée est basée sur deux mesures d'évaluation : (1) distances géographique entre client et Datacenter, (2) la charge du travail de chaque Datacenter. L'algorithme proposé vérifie la disponibilité des ressources sur chaque Datacenter enregistré en terme d'espace de stockage, l'algorithme évalue également le délai d'allocation des centres de données par la somme de la charge de travail avec la distance géographique. Enfin l'architecture proposée est basée sur trois types d'agents : (1) Agent client : qui s'occupe de la formulation de la requête en spécifiant les capacités des ressources que le client cherche, (2) Agent coordinateur : il est responsable de coordination entre : les participants Cloud distribués géographiquement, le client et les fournisseurs de services, afin de trouver le meilleur Datacenter qui réponds au besoin du client (3) Agent contrôleur : responsable de contrôler chaque Datacenter en vérifiant chaque période, les ressources des Datacenter et en envoyant un rapport sur l'état de ces derniers à l' agent coordinateur.

Dans [133] les auteurs ont proposé une approche pour la gestion dynamique et autonome de ressources dans le Cloud, basé sur une architecture distribuée où la gestion des ressources est décomposée en des taches indépendantes et gérée par un ensemble d'agents nœud. Outre, les auteurs proposent l'utilisation de la méthode PROMETHEE à travers une démarche analytique de description Multi Critères exécutée par les agents nœud. Le principal but du système proposé est d'éviter la perte de ressources ainsi que d'éviter la perte de temps lors de la configuration des ressources. L'architecture du système proposée est présentée sous forme d'un réseau d'agents nœuds (AN). Chaque AN s'occupe de la migration de machines virtuelles d'une plateforme à une autre en cas de détection d'anomalies (sur-utilisation ou sous-utilisation). Enfin, la méthode PROMETHEE est utilisée pour choisir la meilleure plateforme de déploiement sur la base des critères de chaque plateforme, ce qui permet également de sélectionner la meilleure machine virtuelle en suivant le même principe.

Dans [134] les auteurs ont défini une approche basée agent pour le contrôle et la supervision de machine virtuelle dans le Cloud, pour la capture d'information sur l'utilisation de la RAM et des CPU. L'architecture du système proposée est basé sur : (1) Interface : qui permet au client de demander et d'accéder au ressources qu'il demande, (2) Agent de gestion de ressources et de MVs : il effectue des rapports sur le taux d'utilisation de la mémoire et des CPU, (3) Agent de supervision de ressources : qui collecte les informations sur l'état des machines virtuelles et les envois à l' administrateur Cloud ,(4) Administrateur Cloud : qui analyse et enregistre les paramètres et capacités des machines virtuelles sur la Dashboard afin de vérifier les performances du système.

Dans [135] les auteurs ont proposé un service basé agent pour l'allocation de ressources dans le Cloud, pour la supervision de fournisseurs de services Cloud afin de les évaluer pour trouver la meilleure configuration de ressources qui répond aux exigences des clients. Le service négocie également un micro-agrément de niveau de services entre clients et fournisseurs, et il supervise également le respect du contrat entre eux. L'architecture proposé par les auteurs est axée sur trois éléments : (1) client : le client doit définir ces exigences en matière de matériel, logiciel, ordre des priorité de ces exigences, ces options, et les dépendances entre les moyennes qu'il demande, en plus le client doit supervisé les ressources Cloud dont il dispose pour informer l'agent d'allocation de la nécessité d'une nouvelle configuration de ressources, (2) fournisseurs de services Cloud : le fournisseur a pour rôle d'offrir une interface standardisée et accessible par l'agent d'allocation afin de faciliter l'émigration et la négociation (négociation d'SLA) entre agents , (3) Agent d'allocation de ressources : l'agent d'allocation est responsable de la découverte des fournisseurs de services, évalue les offres de services, négocie les SLA avec le fournisseur, supervise les violations des SLA, et assiste la migration vers de nouveaux fournisseurs.

Dans [136] les auteurs ont exposé une approche basé agent pour la coordination et la composition de ressources dans le Cloud, afin d'avoir une exécution optimale des taches définies par les applications clients. Le système proposé utilise également un algorithme génétique pour estimer et choisir l'ensemble de ressources optimales pour l'exécution de taches. L'architecture proposée est basé sur 7 élément : (1) Ontologie : pour l'interprétation sémantique des concepts descriptifs des ressources Cloud, (2) services web : pour servir comme interface aux services Cloud ,(3) agent ressources : pour contrôler les services web et pour l'exécution des taches clients ,(4) agent fournisseur de services : pour la coordination , la gestion des agents ressources, la publication des capacités des agents ressources, et pour coordonner avec l'agent intermédiaire sur la base du célèbre protocole contracté net, afin d'allouer les ressources demandées par le client ,(5) agent intermédiaire (broker):pour recevoir la requête du client, estimation des ensembles optimaux de ressources et pour composer les ressources Cloud demandées par le client ,(6) agent consommateur : pour formuler la requête et recevoir ensuite les meilleures offres de services sélectionnées, (7) registre Cloud : ce registre aide les agents fournisseurs et intermédiaires a publié leurs adresses et capacités pour relier les différents composants du système proposé et faciliter la recherche de nouvelles ressources. Enfin les auteurs proposent d'utiliser les chromosomes de l'algorithme génétique afin de représenter le nombre des ressources Cloud à allouer ainsi que le nombre d'heures d'allocation de ces ressources, ensuite une fonction de fitness est calculée pour donner à chaque ressources un niveau de priorité en ordonnant les priorités comme suit : (1) nombre de ressources allouer, (2) cout de ressources, (3) nombre d'instruction (4) respect du temps d'allocation (deadline).

III.5. Conclusion

La découverte et composition de service dans ou hors Cloud reste toujours un problème persistant, notamment avec l'émergence de nouvelles technologies et un défi surtout pour les Plateformes et services orientés Cloud.

Toutefois, il convient de signaler que plusieurs travaux ont été proposés ces 10 dernières années mais aucuns d'eux ne satisfait amplement les exigences de la découverte, sélection composition et trie de services.

Dans ce chapitre nous avons présenté 49 travaux relatifs à l'approche présentée sur cette thèse. Ces approches sont focalisées essentiellement sur deux aspects :

1. Aspect matching : où nous avons présenté des algorithmes de matching pour la découverte et composition de services web et Software-as-a-Service.
2. Aspect architectural : où nous avons présenté des architectures basé agent et basé Cloud pour la découverte et composition de services, modélisation de systèmes Cloud et gestion de ressources dans le Cloud.

Ainsi le prochain chapitre présente notre contribution aux algorithmes de matching pour la découverte, sélection et composition de Software-as-a-Service. Nous nous sommes inspirés des méthodes de matching sémantique, syntaxique et linguistique, pour proposer une solution qui répond à la découverte, sélection et composition de services SaaS atomiques et composites.

Partie II. Contributions

Chapitre IV : SaaS: Découverte et Composition.

Quelle est l'importance de la sélection et composition de services dans le monde réel? Comment décrire un service SaaS? Quel est le processus de découverte et composition de services SaaS? Quel modèle mathématique adopté pour la sélection de services? Comment lier l'étape de sélection à celle de composition? Comment trouver des compromis et combinaisons entre les différents aspects et attributs décrivant des services SaaS? Quel modèle mathématique adopté pour la classification de services SaaS?

Chapitre V : Architecture du système.

Quels sont les objectifs généraux et élémentaires du système proposé? Comment modéliser les composants du système? Comment présenter les paramètres des infrastructures Cloud utilisées? Quel est le modèle d'interaction entre les composants du système proposé? Quels sont les mécanismes de gestion de ressources internes du système? Quels sont les mécanismes de gestion de ressources externes du système?

Chapitre VI : Résultats Expérimentaux.

Quels sont les outils utilisés pour l'implémentation de la base des tests? Quels sont les objectifs des expérimentations sur le prototype implémenté? Quels sont les paramètres d'ajustements des expérimentations? Quelle est la forme du prototype implémenté? Quelles sont les approches utilisées pour la comparaison avec l'approche proposée? Pourquoi ces approches? Quelles sont les mesures utilisées pour l'évaluation de l'approche proposée? Pourquoi l'approche proposée est plus performante que ses semblables?

IV
Contribution 1: Découverte et Composition de SaaS

IV.1. Introduction

Le Cloud computing est un paradigme émergeant qui permet l'auto-approvisionnement de ressources virtuelles et logiques [137]. L'un des modèles les plus importants de livraison dans le Cloud est Software-as-a-Service (SaaS). Un SaaS est un modèle dans lequel le logiciel offert comme service est livré via une interface Web. La découverte des SaaS est l'un des problèmes les plus difficiles et les plus importants sur Internet, en raison de la diversité et la non-standardisation de la description des services Cloud [138, 139]. Il est également très difficile pour les clients de choisir les meilleurs SaaS parmi des centaines, voire des milliers de services découverts [101, 96], car quand les fournisseurs de services Cloud publient leurs caractéristiques et SLA sur leurs sites web, le client doit accéder et comparer manuellement tous les SaaS afin de choisir les meilleurs d'entre eux, ce qui peut être une opération très fastidieuse et fatigante. Un autre problème important est la complexité de la requête du client [107]. Lorsque le client introduit une requête complexe, il est très rare de trouver un SaaS atomique qui satisfait toutes ses exigences, c'est pourquoi, nous devons composer les services sélectionnés afin d'obtenir un SaaS composites virtuels qui satisfait un maximum d'exigences clients.

En Analysant les travaux de recherche existants sur la découverte et composition de service SaaS et Web, nous détectons certaines limites cruciales: la nécessité d'un nouveau modèle pour décrire tous les aspects et parties descriptives d'un SaaS, il n'existe aucune méthode qui combine toutes les parties descriptives d'un SaaS, toutes les méthodes séparant entre les attributs fonctionnels et non-fonctionnels lors de la création des services composites, et la requête est complètement ignorée lors de la réalisation des tests de connectivité pour la création de services composites, ce qui peut réduire considérablement l'efficacité des services retournées au client, et finalement, il n'y a aucun système complet qui traite la découverte, la sélection, la composition et la classification des services SaaS atomiques et composites.

Afin de répondre aux limitations citées ci-dessus, nous proposons un système complet qui traite la découverte, la sélection, la composition et la classification des services SaaS dans le Cloud, en réalisant les objectifs suivants :

- Utilisation du standard WSDL pour décrire les services SaaS, car la majorité des entreprises sur Internet utilisent des langages de description de services Web comme WSDL, OWL- S, et WSMO ... etc.[140] pour déployer leur Software-as-a-Service, car les SaaS peuvent être décrits en utilisant les mêmes paramètres que les services Web tels que : les entrées, les sorties, le nom du service, l'emplacement du service ... etc.
- Développer un modèle de description complet qui prend en considération toutes les caractéristiques descriptives des services SaaS.

- Présentation d'un nouvel algorithme de matching pour la découverte et la sélection de services SaaS.
- Présentation d'un nouveau algorithme de matching pour la composition et le trie de services SaaS.
- Combinaison des attributs fonctionnels et non-fonctionnels pour la composition de services SaaS.
- Combinaison des attributs fonctionnels, non-fonctionnels et Cloud pour le tri des services SaaS.

Ce chapitre est organisé comme suit :

Dans la section 2 nous présentons un exemple descriptif de l'énoncé du problème traité sur ce chapitre. Dans la section 3, nous présentons le modèle de description des services SaaS proposé. Dans la section 4 nous présentons l'architecture globale du modèle proposé. Dans la section 5, nous présentons l'algorithme de découverte et sélection de services SaaS. Dans la section 6, nous présentons l'algorithme de composition et trie de services SaaS. Dans la section 7, nous présentons les complexités temporelles des algorithmes de matching proposés. Enfin, la section 8 est la conclusion de ce travaille.

IV.2. Enoncé du problème

Prenons le scenario d'une entreprise qui a besoin d'un système d'archivage de document, situé en Algérie. Une telle entreprise nécessite un Service complexe composé de 5 Type de SaaS :

1. SaaS pour la gestion de base de données: Ajout, suppression, modification et catégorisation de documents.
2. SaaS pour superviser le transfert de fichiers : échanges de données B2B et A2A (inter-applications).
3. SaaS pour l'intégration d'applications: orchestration et intégration des applications de l'entreprise demandeuse au sein du système d'archivage.
4. SaaS pour la gestion de la logistique entre entreprises partenaires et clients: livraison, facturation, traitement de commandes.
5. SaaS pour la sécurisation d'accès aux documents: authentification, enregistrement de traces du personnel (en cas d'abus d'utilisation), protection du trafic entrée/sortie, prévenir les pertes de données, vérification de la conformité avec les règlements de l'entreprise, Anti-virus.

Afin d'assurer un bon service d'archivage les SaaS doivent être composé et ordonnancer de la manière suivante :

1. chaque employé responsable de l'opération d'archivage de l'entreprise demandeuse, utilise son identifiant et mot de passe pour accéder à l'espace d'archivage par l'intermédiaire d'une Interface offerte par le SaaS qui est responsable de l sécurisation des données.
2. Le SaaS responsable de sécurisation des données est composé de trois types de SaaS, ce qui permet aux employés d'accéder à trois types d'Interface, selon l'opération que cet employé cherche à faire :

- Si l'employé cherche à effectuer des modifications sur les documents archivés, il utilise l'interface offerte par le SaaS responsable de la gestion de la base de données, qui est composée avec le SaaS responsable du transfert de fichier.
- Si l'employé cherche à relier des applications propres à son entreprise avec celle offertes par le système d'archivage, il utilise l'interface offerte par le SaaS responsable de l'intégration d'applications.
- Si l'employé cherche à traiter des commandes extérieures à son entreprise, il utilise l'interface offerte par le SaaS responsable de la gestion de la logistique.

Afin d'assurer une bonne qualité de service, et afin de respecter la vie privé de ces clients, l'entreprise demandeuse exige que les Datacenter de stockage des données soient situés en Algérie et que tous les SaaS cités ci-dessus soit menus d'une SLA. Outre, le premier et deuxième SaaS doivent avoir un degré de disponibilité excellent, puisque il s'agit de gérer de grandes quantités de données, et le troisième et quatrième SaaS doivent avoir un excellent degré de performance puisque il s'agit de gérer des applications, enfin, le dernier SaaS doit avoir un excellent degré d'intégrité puisque il s'agit de faire face à des menaces sécuritaires continues.

Au lieu que l'entreprise demandeuse développe ces propre SaaS, ce qui nécessite beaucoup de temps et d'argents, cette dernière peut utiliser des services déjà disponibles sur Internet, afin d'assurer cela, deux mécanisme sont nécessaires**:** (1) Mécanisme de découverte et sélection de SaaS, (2) Mécanisme de composition et classification de SaaS.

Dans ce travail on utilise un nouveau modèle de matching pour la sélection, composition et classification de services et qui prend en considération l'aspect fonctionnel, qualité, et Cloud des services SaaS. L'approche proposée combine ces trois aspects lors de l'exécution de l'algorithme de matching de telle sorte que la composition devient une étape naturelle et nécessaire après l'étape de découverte.

Enfin, nous supposant que les capacités des ressources sur lesquelles sont déployés les SaaS, dépassent largement les besoins des clients, donc la capacité ne représente plus une contrainte.

IV.3. Description des services SaaS

Le Cloud computing peut être vue comme un entrepôt de services qui peuvent être exploités et utilisés pour diverses fins. Ainsi, l'un des problèmes les plus importants est de connaitre la meilleure méthode pour sélectionner et composer les meilleurs services qui répondent aux exigences d'entreprises et de clients particuliers (demandeurs de services). Pour résoudre ce problème, nous pouvons utiliser un moteur de recherche et de composition de service. Mais, ce type de moteurs a besoin d'une description complète des services Cloud, ce qui lui permettra de découvrir et composer les meilleurs services. Pour avoir une description complète d'un Software-as-a-Service nous avons besoin de décrire au moins quatre parties:

- Partie de catégorisation: la partie de catégorisation permet de répondre aux questions suivantes: quelle est le domaine du service ? quelle est son emplacement? Les paramètres utilisés dans cette partie sont : Nom du service, nom du site web hébergeant le service, et les commentaires des développeurs (introduit dans la balise « documentation »).

- Partie fonctionnelle : la partie fonctionnelle répond aux questions suivantes: quelle est la fonction générale du service? quelles sont exactement les fonctions techniques de chaque opération effectuée par le service? Les paramètres utilisés dans cette partie sont: le nom du portType, les noms des opérations, les noms des entrées/sorties.
- Partie non-fonctionnelle: la partie non-fonctionnelle répond à la question suivante: quel est le niveau de qualité offerte par le service? Beaucoup de paramètres sont utilisés dans cette partie, par exemple: temps de réponse, disponibilité, fiabilité, transparence ... etc.
- Partie Cloud: la partie Cloud répond à la question suivante: quelles sont les caractéristiques industrielles et économiques fixées par le fournisseur de services? Après avoir étudier les références suivantes : [95, 141, 142, 139, 101], nous avons recueilli 24 caractéristiques Cloud qui représentent les attributs les plus répétés et les plus importants de notre point de vue. Ensuite, nous les avons classés en trois grandes catégories: catégorie des valeurs numériques, catégorie des valeurs textuelles, et catégorie des valeurs prédéterminées. La classification proposée des caractéristiques Cloud est basée sur la nature des valeurs que prend chaque caractéristique, car chaque type de valeur (numérique, textuelle, et prédéterminée) doit avoir sa propre équation qui permet de calculer le degré de similarité entre les caractéristiques Cloud de chaque service et les exigences du client. Le tableau suivant présente les caractéristiques Cloud collectées ainsi que leurs catégories :

Catégories	**Caractéristiques**	**Description / Valeur**
Valeur numérique	Nombre de clients	Entier
	Valeur de la marque du vendeur	Réel
	Version	Convertie en entier
	Expérience des ingénieurs	Entier (nombre d'année)
	Niveaux de compétence	Entier
Valeur textuel	Contributeur	C'est le nom de l'Organisation qui contribue au développement de nouvelles versions.
	Créateur	C'est le nom de l'Organisation qui a créé le SaaS
	Possesseur	C'est le nom de l'Organisation qui possède le SaaS
	Publicateur	C'est le nom de l'Organisation qui publie le SaaS.
	Emplacement	C'est le nom du pays où les Datacenter qui hébergent le SaaS sont déployés.
	Identificateur	C'est une combinaison de chiffre et lettre qui permet d'identifier le SaaS de façon unique.
	Langues supporté par le service	C'est la liste des langues supportées par le SaaS

	Adhérassions aux Standards	C'est la liste des standards (ex. WSDL, OVF, TOSCA ... etc.) utilisés pour développer le SaaS.
	Certification	C'est le(s) nom(s) du certificat(s) gagné par le SaaS, attestant la conformité à certains standards ou lois.
Valeur Prédéterminé	Système de paiement	(i) Free (ii) Dynamic (iii) Pay per use
	Déploiement sur le Cloud	(i) Private (ii) Public (iii) Hybrid (iv) Community
	Degré d'ouverture cloud	(i) Unknown_Limited (ii) Basic (iii) Moderate (iv) Complete
	Standardisation	(i) No (ii) Public API (iii) Part (Standard, Organization)
	Contrat formel	(i) No SLA (ii) SLA
	Type de la licence	(i) Open source (ii) Propriety
	Intégration	(i) Yes (ii) No
	Support pour appareil mobile	(i) Yes (ii) No
	Support pour fonctionnement Hors-ligne	(i) Yes (ii) No
	Sécurité externe	(i) None (ii) SSL (iii) PKI (iv) VPNs

Tableau IV.1: Résumé des caractéristiques Cloud.

Enfin, la figure suivante modélise le problème de découverte et composition de Software-as-a-Service:

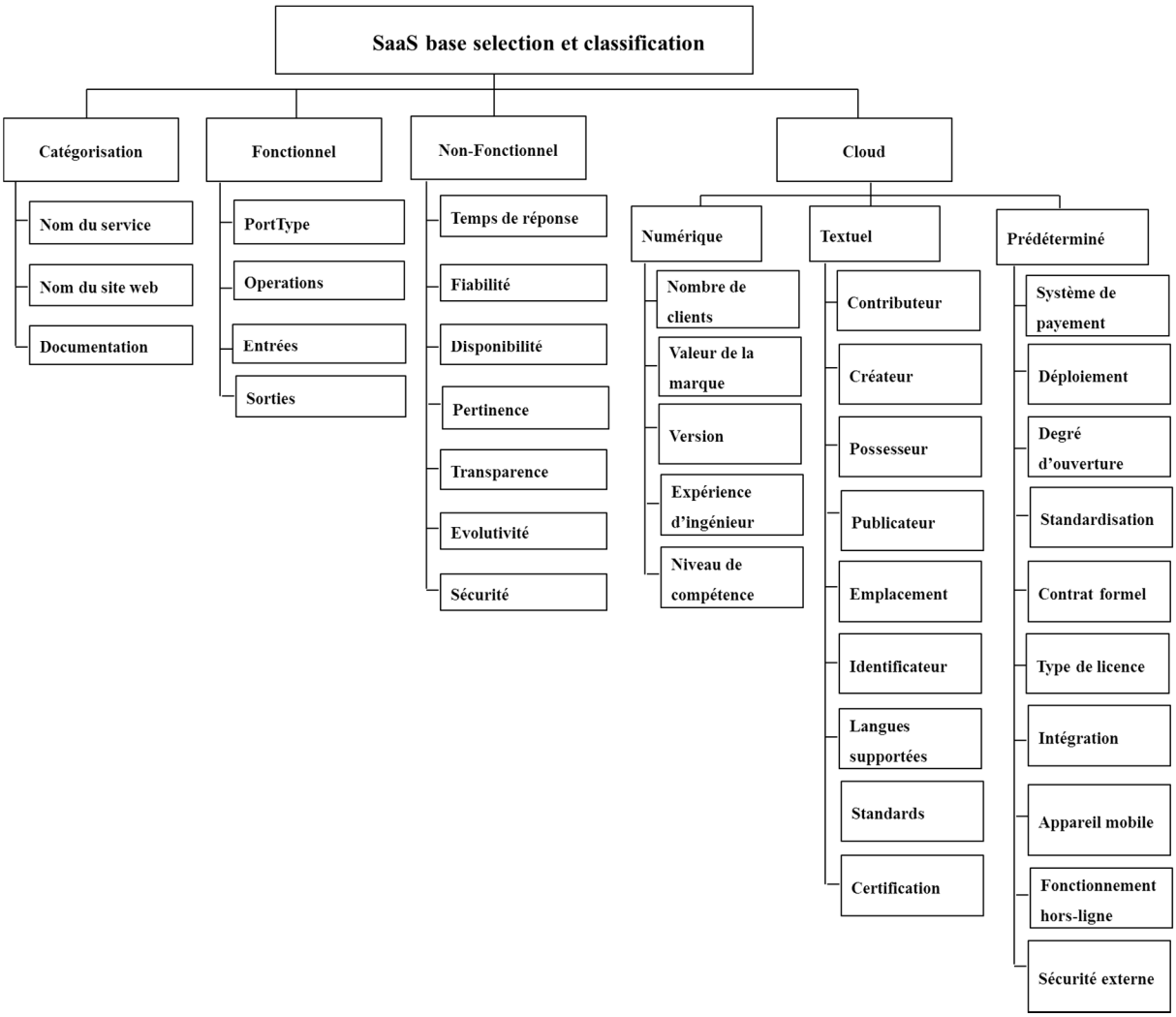

Figure IV.1: Représentation hiérarchique des attributs SaaS.

IV.4. Architecture du modèle proposé

Le système de découverte et composition proposé est composé de trois principaux sous-systèmes: (i) Service d'enregistrement, (ii) Service de découverte et de sélection, (iii) Service de composition et classification. L'architecture du système proposé est représentée par la figure IV.2 :

IV.4.1. Services d'enregistrement

Ce sous-système est composé de deux modules : module de réception et module d'indexation.

- Module de réception: ce module permet la réception des services SaaS, en recevant : les fichiers WSDL, les valeurs de qualité de service et les caractéristiques Cloud, afin d'envoyer les fichiers WSDL ainsi que les valeurs QoS et Cloud au module d'indexation pour les regrouper en plusieurs cluster. Enfin, le module de réception est responsable de l'acceptation des mises à jour des services enregistrés.

- Module d'indexation: ce module est responsable du regroupement des services publiés sur des clusters sur la base de leurs paramètres de catégorisation et de fonctionnement. A chaque fois, un nouveau service est reçu, le module d'indexation calculer le degré de similarité entre lui et la similarité moyenne de chaque cluster [143], ensuite le nouveau service est affecté au cluster avec le meilleur degré de similarité. Après la fin de l'opération de regroupement, le module d'indexation crée et stocke l'arbre WSDL de chaque service reçu.

IV.4.2. Service de découverte et sélection

Ce sous-système est composé de trois modules: module de prétraitement des requêtes, modules de matching, et modules de préparation à la composition.

- Module de prétraitement de la requête: ce module est responsable de la réception de la requête du client (en langage naturel) et les poids des caractéristiques QoS et Cloud. Ce module est également responsable de la simplification de la forme de la requête en utilisant les techniques de Tokenisation et lemmatisation.
- Module de matching: ce module est responsable du calcul du degré de matching entre la requête du client et les signatures de chaque cluster, afin de choisir un ensemble de services qui correspondent syntaxiquement aux exigences du client. En outre, le module de matching est responsable de ce que nous appelons, le degré d'existence global de la requête du client dans les arbres WSDL des services choisis, afin de sélectionner un ensemble de services qui correspondent sémantiquement aux exigences du client.
- Module de préparation à la composition : ce module est responsable de l'affectation de chaque service sélectionné à sa classe. L'algorithme de découverte et de sélection proposée utilise trois classes: classe d'existence complète, classe d'existence partielle, et classe d'existence élémentaire (section 5.4). Ces classes vont nous aider à faciliter l'opération de composition de services.

IV.4.3. Service de composition et classification

Ce sous-système est composé de quatre modules : module de calcul des scores QoS, module de catégorisation, module des tests de connectivités, et module de Classification.

- Module de calcul des scores QoS: ce module est chargé de calculer les scores de qualité des services atomiques sélectionnés sur la base de la technique additive de pondération simple (section 6.1). En outre, ce module est responsable de calculer les scores de qualité des services composites, afin de permettre au module de classification de les classer.
- Module de catégorisation : ce module est responsable de la création de deux tables, la première table contient les services classés selon leur degré d'existence, et la seconde table contient les services classés selon leurs scores QoS. En outre, les services sont classés selon le nombre de mots clés de la requête dans chaque fichier WSDL (le nombre de classes est le nombre de toutes les combinaisons possibles entre les mots-clés de la requête) (section 6.2).
- Module des tests de connectivité: ce module est responsable d'effectuer les tests de connectivités entre les services sélectionnés, en suivant quelques règles (Section 6.3.1) et en utilisant les tables de catégorisation afin de choisir alternativement (un

service de la première table avec un autre service de la deuxième table) les services sur lesquels les tests sont effectués. En outre, ce module utilise un nouveau modèle de coûts (section 6.3.2) pour calculer la distance d'édition entre les arbres WSDL, afin de trouver les meilleures chaînes de connectivités, qui seront retournées au client comme SaaS composites.

- Module de Classification: ce module est responsable de la classification des services atomiques et/ou composites. Le module de classification combine les caractéristiques Cloud, QoS et le degré d'existence pour classer les services sélectionnés.

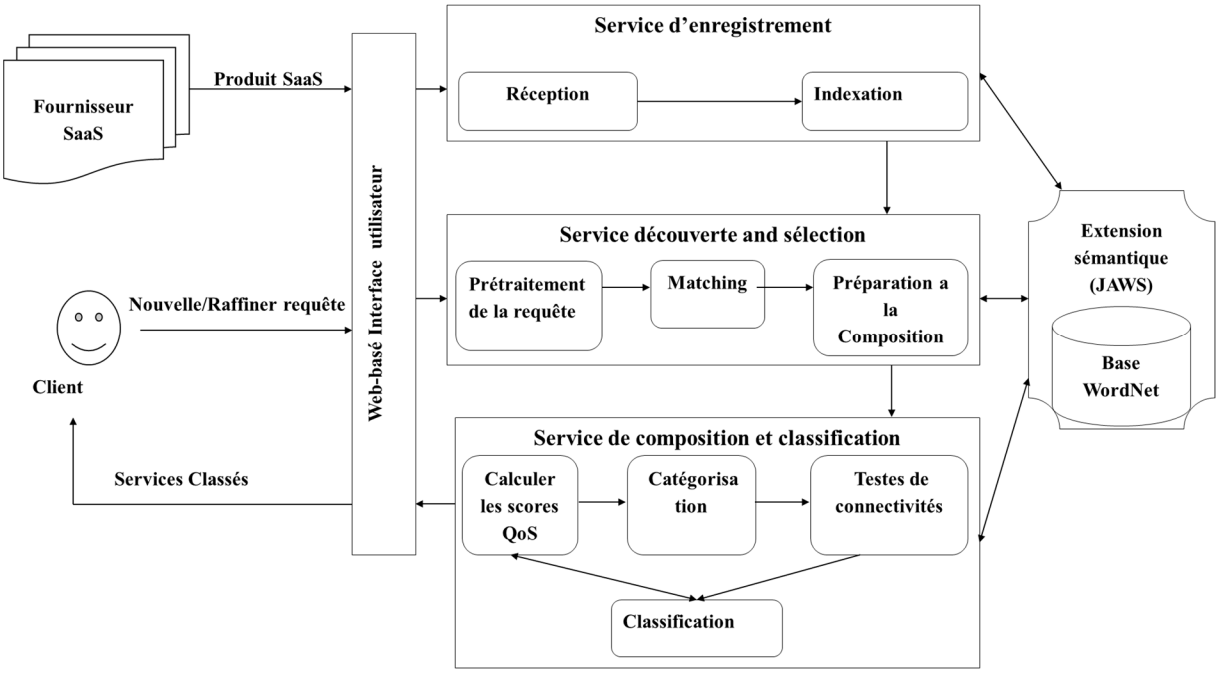

Figure IV.2: Architecture du système proposé.

IV.5. Processus de découverte et sélection

L'algorithme de découverte et sélection est basé sur le calcul du degré d'existence (pas de similarité) entre la requête du client (exprimé en langage naturel) et les arbres WSDL des services indexés. L'idée consiste à décomposer les arbres WSDL en un ensemble de sous arbres, selon la signification sémantique de chaque élément WSDL. Puis on calcule le degré de correspondance entre mots clés de la requête avec les mots clés de chaque sous- arbre. Les résultats donnent le degré d'existence structurale de la requête dans chaque sous- arbre WSDL. Puis nous calculons le degré d'existence global entre la requête et l'arbre WSDL par la somme des degrés d'existence structurale des sous-arbres divisées par le nombre des sous-arbres.

Le mécanisme de découverte et sélection est basé sur quatre étapes principales :

IV.5.1. Création d'arbres WSDL

A Chaque fois qu'un nouveau fichier WSDL est collecté, le mécanisme de découverte et sélection de services crée l'arbre qui lui correspond, en exécutant les étapes suivantes:

IV.5.1.1. Règles de création des arbres WSDL

Pour avoir une représentation complète des différents éléments WSDL, nous créons l'arbre en suivant quelques règles qui décrivent les nœuds de chaque sous arbre WSDL:

1. L'élément racine de l'arbre WSDL est la valeur de la propriété « name » de l'élément « définition ». Par exemple : <wsdl : definition name = " ***PurchaseOrder*** " />
2. Le premier enfant de la racine est la valeur de la propriété « name » de l'élément « service ». Par exemple, <wsdl : service name = " ***PurchaseOrderService*** " />
   - Le premier enfant du nœud service-name est la description textuelle du service. Par exemple : <wsdl:documentation>***on-lignepurchase and order by criditcard***</ documentation>
   - Le deuxième enfant du nœud service-name est le nom du site Web du service. Par exemple : < soap: adresse location = " http:// ***IPurchaseOrder.org*** " />.
3. Le deuxième enfant de la racine est la valeur de la propriété « name » de l'élément « Binding ». Par exemple : <wsdl:bindingname="***PurchaseOrderBinding***" />
4. Le troisième enfant de la racine est la valeur de la propriété « name » de l'élément « portType »(en WSDL 2.0 le terme « portType » est remplacé par le terme « interface »), Par exemple : <wsdl : portTypename = " ***IPurchaseOrder*** " />
   - Les enfants du nœud portType-name sont les valeurs de la propriété « name » de chaque élément « operation ». Par exemple: <wsdl: operation name= "***Order***" />, <wsdl: operation name="***OrderStatus***">.
     - Les enfants du nœud operations-name sont les valeurs de la propriété « name » des éléments « entrées » et « sorties ». Par exemple : <wsdl:output message="tns:***GetOrderStatusResponse***" />.
5. Le dernier enfant de la racine est composé des descriptions textuelles que nous trouvons dans les différentes balises « documentation » (ce genre de balises sert a documenter le fichier WSDL mais il est facultatif).

IV.5.1.2. Enrichissement sémantique des arbres WSDL

Tout fichier WSDL est associé à un schéma XML qui sert à valider et déterminer les types des données du document WSDL. Les types de données qu'on trouve sur un schéma XML sont divisés en deux catégories : les types de données primitives comme *int* ou *bool* et les types de données complexe comme <xsd: elementmin Occurs = " 0 " name = " ***expectedShipDate***" type = " xsd: string " / >, le principal avantage des types complexes est qu'ils donnent une description sémantique des valeurs de chaque propriété des éléments WSDL. C'est pourquoi il est très recommandé de transformer les types des données primitives à des types de données complexes.

Pour enrichir sémantiquement l'arbre WSDL on ajoute aux nœuds entrées et sorties, les valeurs de la propriété « name » dans les types des données complexes qui leurs correspondent sur le schéma XML.Par exemple, nous ajoutons: <xsd: elementminOccurs=nom"0" = "***quantité***" type = "xsd: int" />et<xsd: elementminOccurs= "0"name =Type"***productName***" = "xsd: string" /> au nœud qui correspond à l'entrée: <wsdl: Input message ="tns: ***PurchaseOrder***">.

Enfin, l'arbre WSDL est traversé en profondeur, afin : d'élargir les abréviations et acronymes à leurs formes originales, supprimer les mots qui ne véhiculent aucun sens (stop words en anglais), et lemmatiser les mots-clés.

IV.5.1.3. Définitions des sous arbres

Un arbre WSDL est structuré en 5 types de sous arbres, chaque type de structure représente une partie du fichier WSDL et chaque type se compose d'un ou plusieurs nœuds :

1. Sous arbre Racine: il contient, seulement, la valeur de la propriété « name » de l'élément « definition »
2. Sous arbre Service: il contient trois nœuds : la valeur de la propriété « name » de l'élément « Service », la description textuelle du service qu'on trouve dans la balise « documentation » et le nom du site web du service.
3. Sous-arbre liaison: il contient, seulement, la valeur de la propriété « name » de l'élément « Binding ».
4. Sous arbre Globale PortType: ce sous arbre est composé de deux sous arbres :
   - Sous-arbre opérations: il est composé des valeurs de la propriété « name » des éléments « operation », « Input » et « «output » de chaque opération.
   - Sous-arbre portType: il contient, seulement, la valeur de la propriété « name » de l'élément « protType ».
5. Sous arbre documentation: le nombre des nœuds du sous arbre« ducumentation » est égale au nombre de balises « documentation » utilisé par le développeur WSDL afin de commenter son fichier.

IV.5.1.4. Prétraitements

Avant tout traitement, la requête du client comme les arbres de WSDL doit être bien préparée pour calculer le degré d'existence globale, cela se traduit en trois principales étapes:

IV.5.1.4.1. Extraction de Mots clés de la requête

Nous éliminons tout simplement les mots qui ne véhiculent aucun sens de la requête, et on ne maintien que les mots-clés, parce que ces dernier sont les seuls éléments qui exprime les besoins du client.

IV.5.1.4.2. Normalisation de la Requête

Pour simplifier les formes des mots clés de la requête et des arbres WSDL, nous appliquons quelques techniques de traitement de langages Naturel:

1. **Tokenisation** : le but de la tokenisation et d'arriver à distinguer et extraire le sens de base de chaque mot-clé (dans la requête du client comme dans les arbres WSDL), afin d'arriver à cela, on applique des opérations telles que: La segmentation, traitement des séparateurs ambiguës (-) sauf les mots composés, traduire les nombres en mots ... etc.
2. **Lemmatisation** : En utilisant de la base lexical WordNet, on remplace chaque mot clé de la requête par son « lemme » pour faciliter le rapprochement sémantique et lexicale entre les mots-clés des arbres WSDL et leurs synonymes (éventuels) dans la

requête du client. L'avantage de la lemmatisation par rapport à la transformation des mots à leur racine (Word Stemming en anglais) (qui est largement utilisée dans les algorithmes d'extraction d'information) est que la lemmatisation préserve la sémantique et le contexte des mots lemmatisé dans leur expression [144].

IV.5.1.4.3. Substitution

Pour rapprocher la requête et les arbres WSDL, ces derniers sont traversés en profondeur, afin de remplacer chaque mot-clé dans les arbres WSDL par son synonyme dans la requête, en utilisant la base lexicale WordNet. Dans le jargon WordNet on dit que deux termes différents sont synonymes s'ils désignent le même sens, ou s'ils sont interchangeables dans des contextes spécifiques.

IV.5.2. Recherche basée Mots-clés

Lorsque le client introduit sa requête, le mécanisme de découverte et sélection extrait de cette dernière, les mots-clés et leurs synonymes, et lance une recherche basée sur ces mots-clés, en utilisant un moteur de recherche telle que Lucene [145]. Les résultats représentent les arbres WSDL qui correspondent syntaxiquement à la requête du client. Nous utilisons la recherche syntaxique parce que la base des services contient des milliers de fichiers WSDL, alors que si nous essayons de calculer directement le degré d'existence globale entre chaque arbre WSDL et la requête du client, cela va consommer beaucoup de temps et d'espace mémoire. En utilisant un moteur telle que Lucene avec son algorithme de recherche et d'indexation basée mots clés, nous réduisons le nombre des Arbres WSDL pour lesquelles nous calculons le degré d'existence, cela implique la réduction: de la consommation de l'espace mémoire, destiné à stocker les arbres WSDL associés à chaque requête du CPU destinée à calculer le degré d'existence globale, et enfin le temps de réponse globale du moteur proposé.

IV.5.3.Calcul du degré d'existence globale

On se base sur la base lexicale WordNet d'un côté et d'un autre côté sur la mesure de similarité sémantique présentée dans[146] (qui permet d'obtenir 82% de corrélation avec les résultats obtenus par des êtres humains),et cela pour mesurer le degré de similarité entre les mots-clés de la requête et le nœud de chaque arbre WSDL. Ensuite on utilise les résultats des calculs des degrés de similarités, pour calculer le degré d'existence structurale de chaque sous arbre, ce qui nous permet, par la suite, de calculer le degré d'existence globale entre la requête et les arbres WSDL.

IV.5.3.1.Degré d'existence Structurale

L'utilisation du modèle de représentation d'espace vectoriel (Vectorial Space Model en anglais (SVM)), permet de calculer, syntaxiquement, les poids-termes des mots-clés d'un arbre WSDL, en utilisant la formule suivante:

$$w_{T,j} = Tf_j * \log\left(\frac{D}{Df_j}\right) \quad (1)$$

Où:

Tf_j: représente la fréquence du terme j dans le nœud de l'arbre WSDL T.

Df_j: représente le nombre de nœuds, de la même catégorie, qui contiennent le terme *j*.
D : représente le nombre d'arbres WSDL.

Cette même formule permet également de calculer les poids-termes des mots-clés la requête, en utilisant la formule:

$$w_{Q,i} = Qf_i * \log\left(\frac{1}{Df_i}\right) \quad (2)$$

Où:
Qf_i: représente la fréquence du terme *i* dans la requête *Q*.

Ce qui permet de calculé le degré de similarité entre les mots-clés de la requête et ceux d'un arbre WSDL, en utilisant la formule suivante:

$$sim(Q, T_r) = \frac{\sum_i (w_{Q,i} * w_{T_r,j})}{\sqrt{\sum_i (w_{Q,i})^2} + \sqrt{\sum_j (w_{T_r,j})^2}} \quad (3)$$

Où :
T_r : représente l'arbre WSDL *r*.

Mais le modèle VSM ne comporte aucune représentation sémantique des Mots-clés de la requête ou des arbres WSDL (hyponymes et hyperonymes), c'est pourquoi nous utilisons une mesure de similarité sémantique basée sur WordNet, pour calculer le degré d'existence entre la requête du client et les arbres WSDL. Le calcul du degré d'existence structurale de chaque sous arbre, suit 4 étapes:

1. Repondération des termes: pour chaque terme d'une requête *i* nous utilisons les termes sémantiquement similaires et qui appartiennent au même vecteur, pour ajuster les poids-termes des mots-clés de la requête:

$$w_{Q,i} = w_{Q,i} + \sum_{sim(i,j)\geq t}^{j\neq i} w_{Q,i} sim(i, j) \quad (4)$$

Où:
t : représente un seuil prédéfini (*t* = 0,6).
j: représente les termes similaires aux termes i dans le même vecteur.

2. Enrichissement sémantique des termes: l'enrichissement sémantique est l'opération qui permet d'augmenter les termes de la requête, en ajoutant à chaque mot clés ces synonymes, hyponymes et hyperonymes qui sont issus de la base lexicale WordNet. Ensuite ces termes, liés sémantiquement à la requête, sont ajoutés à la représentation de la requête. Puis on applique la formule suivante pour calculer le poids-terme de chaque mot clé de la requête:

$$w_{Q,i} = \begin{cases} \sum_{sim(i,j)\geq T}^{i\neq j} \frac{1}{n} w_{Q,i} sim(i, j) & \text{si } i \text{ est un nouveau terme} \\ w_{Q,i} + \sum_{sim(i,j)\geq T}^{i\neq j} \frac{1}{n} w_{Q,i} sim(i, j) & \text{si } i \text{ a un poid } w_{Q,i} \end{cases} \quad (5)$$

Où:

t: représente un seuil prédéfini (*t*=0,45).

n: est le nombre d'hyponymes pour chaque terme enrichi *j*.

3. Similarité entre requête et nœud: après la repondération et l'enrichissement sémantique des termes de la requête, on calcule le degré de similarité entre la requête du client et chaque nœud de l'arbre WSDL, selon la formule suivante:

$$sim\ (Q, N) = \frac{\sum_i \sum_j w_{Q,i} w_{T,i} sim\ (i, j)}{\sum_i \sum_j w_{Q,i} w_{T,j}} \quad (6)$$

Où:

N: représente le nœud cible.

i: représente un terme de la requête *Q*.

j: représente un terme du nœud *N*.

$w_{T,j}$: représente le poids d'un terme du nœud *N*, qui est calculé en utilisant le modèle de représentation vectoriel présenté ci-dessus.

4. Calcul du degré d'existence structurel: après le calcul du degré de similarité entre chaque nœud de l'arbre et la requête, on calcule le degré d'existence structure le pour chaque catégorie de sous arbre, selon la formule suivante:

$$Exis\ (Q, ST) = \frac{\sum_{i=1}^{i=Nbn} sim\ (Q, N_i)}{Nbn} \quad (7)$$

Où:

ST: représente la sous arbre cible.

Nbn: représente le nombre de nœuds du sous arbre cible.

N_i: représente le nœud *i* du sous arbre cible.

IV.5.3.2. Degré d'existence global

Le degré d'existence globale de la requête du client dans un arbre WSDL est calculé par la formule suivante :

$$Glob\_Exis(Q,T) = \frac{\sum_{i=1}^{i=Nbs} Exis(Q, ST_i)}{Nbs} \quad (8)$$

Où:

T: représente un arbre WSDL.

Nbs : représente le nombre de sous arbres.

Explication 1 : la première et principale question qui vient à l'esprit à propos du mécanisme de découverte et composition proposée est : pourquoi nous utilisons le mot degré d'existence au lieu de degré de similarité ? Le mot similarité est utilisé lorsqu'on compare deux choses qui appartiennent au même type ou qui ont la même forme, par exemple, comparaison entre deux arbres, comparaison entre deux textes ... etc. Avec le mécanisme de découverte et composition de service qu'on propose, on compare entre la requête du client,

qui est exprimée en langage naturel et des fichiers WSDL, qui sont représentés sous forme arborescente, c'est pour cela que nous utilisons le mot existence à la place de similarité. L'idée consiste à maintenir la requête du client sous la forme dans laquelle elle a était saisie (langage naturel), cela nous permet : d'éviter la perte sémantique d'une partie de la requête, d'optimiser le temps de réponse du système (puisque on n'est pas obligé de passé par une étape de réécriture de la requête), et d'utiliser les mots-clés de la requête comme guide pour la composition des services (voir section catégorisation ci-dessous).

IV.5.4. Sélection de services et Préparation à la composition

Le mécanisme de découverte et sélection de service choisit les services qui ont le plus haut degré d'existence, en associant, à chaque service, une description textuelle basée sur le nombre des mots clés (ou leurs synonymes sémantiques) de la requête trouvée dans chaque arbre WSDL :

- Si tous les mots clés de la requête ont leurs synonymes dans les arbres WSDL, nous dirons que nous avons une ***existence complète*** de la requête du client dans le fichier WSDL correspondant.
- Si tous les mots-clés des sous-arbres « operation » ont leurs synonymes dans une partie de la requête, nous dirons que nous avons une ***existence partielle*** de la requête du client dans le fichier WSDL correspondant. Outre, nous distinguant deux cas particuliers :
  a) Si tous les mots-clés des nœuds qui représentent les « sorties » du service, ont leurs synonymes dans une partie des mots-clés de la requête, nous dirons que nous avons une ***existence partielle en sortie*** de la requête du client dans le fichier WSDL.
  b) Si tous les mots clés des nœuds qui représentent les « entrées » du service, ont leurs synonymes dans une partie des mots clés de la requête, nous dirons que nous avons une ***existence partielle en entrée*** de la requête du client dans le Fichier WSDL.
- Si une partie des mots clés de la requête ont leurs synonymes dans les sous-arbres, nous dirons que nous avons une ***existence élémentaire*** de la requête du client dans le fichier WSDL correspondant.

Algorithme 1 Mécanisme de découverte et sélection

Input: (i) Requête du client ***Q*** (ii) Arbre WSDL Enrichi***EWT***
Output: (i) Services sélectionnés (ii) Dégréd'existence global
// Prétraitement
1: Extraction de mots clés, Tokenisation, Lemmatisation (***Q***);
2: ***EWT*** = Recherche Syntaxique (***Q***, ***EWT***);
3: Substitution (***Q***, ***EWT***);
// Dégré d'existence globale
4: ***Q***=Re-Pondération de Mot Clés (***Q***);
5: ***Q*** = Enrichment Semantique (***Q***);
6: **Foreach*EWT*do**
7: d = GetDepth (***EWT***);
8: **For** (i=d to i=0; i--) **do**

```
9:      Foreach nœud  avec le même identifient do // chaque sous arbre WSDL a son propre
                                                    identifient 'id'
10:       Exis_Structural [id] = Exis_Structural [id] + Similarité_Semantique (Nœud [id], Q);
11:     Exis_Structural [id] =Exis_Structural [id] / Number_Noued [id];
12:     For (i=1 to i=id; i++)
13:       Exis_Global = Exis_Global +Exis_Structural [i];
14:       Exis_Global = Exis_Global / Nombre_Sous_Arbre
// Preparation a la composition
15:   For (i=d to i=0; i--) do
16:     ForeachNœuddo
17:       NbKWN = Nombre des mots clés de sur chaque nœud;
18:       NbKWNOp = Nombre de mots clés sur chaque nœud des sous arbres opérations;
19:     if (NbKWN est égale au nombre de mots de la requête) then
20:       Return: Existence complète;
21:     else
22:       if (NbKWNOp est égale au nombre de mots de la requête) then
23:         Return: Existence partiel;
24:       else
25:           Return: Existence Elémentaire;
```

Algorithme IV.1 : Mécanisme de découverte et sélection de services SaaS

Après la fin de cette étape, le mécanisme de composition et classification reçoit : les sous arbres « portType » de chaque service sélectionné, la description textuelle des services, le degré d'existence globale, et les mots clés de la requête ainsi que leurs distributions dans les sous arbres « «portType ».

IV.6. Processus de composition et classification

Le mécanisme de composition et classification est basé sur une nouvelle méthode de catégorisation de services, qui consiste à créer une catégorie pour chaque ensemble de services qui partage des caractéristiques commune par rapport à la requête, afin de déterminer les différents liens de connectivité entre les services sélectionnés. L'idée consiste à créer un service virtuels composites qui contient un maximum de mots-clés de la requête, puisque l'idée de base consiste à utiliser la requête comme guide pour l'exécution de chaque étape du mécanisme, afin de garantir que ce dernier ne dévie jamais des besoins du client (exprimés dans sa requête), ce qui implique que les services qui contiennent un maximum de mots-clés de la requête sont les services qui répondent le mieux aux exigences fonctionnelles des clients. Outre, la catégorisation permet également de combiner entre aspect fonctionnel et non fonctionnel lors de l'exécution du mécanisme (on ne laisse pas l'aspect fonctionnel à la fin, comme le font toutes les méthodes de découverte et composition de service, vu l'importance de cette aspect dans le cadre du Cloud), dans le but de trouver un compromis entre ces deux aspects.

IV.6.1.Pré-classification de Services basée QoS

On utilise la technique de Pondération additive simple (Simple Additive Weighting (SAW) en anglais) pour modéliser le problème de classification de services basés sur leurs qualités. Le principe de cette Technique SAW consiste en la création, d'un vecteur de valeurs

qui représentent les critères de qualité de chaque service. La technique SAW est basée sur deux phases:

1. Phase de normalisation: afin d'unifier les critères QoS, nous utilisons deux formules de normalisation qui traitent : les critères en croissances (*ex.* évolutivité, fiabilité, robustesse, flexibilité ... etc.) et les critères décroissants (*ex.* prix, temps de réponse, taux d'erreur...etc.):

$$N_{i,j} = \begin{cases} \dfrac{\max_i(NN_{i,j}) - NN_{i,j}}{\max_i(NN_{i,j}) - \min(NN_{i,j})} & \text{Critaires QoS a reduire.} \\ \dfrac{NN_{i,j} - \min_i(NN_{i,j})}{\max_i(NN_{i,j}) - \min(NN_{i,j})} & \text{Critaires QoS a augmenter.} \end{cases} \quad (9)$$

Où:

$N_{i,j}$: représente la valeur de qualité de service *i* normalisée à partir de *j* services découverts.

$NN_{i,j}$: représente la valeur de qualité de service i non normalisée à partir de *j* services découverts.

2. Pondération et calcul de score: selon la technique SAW, le client définit les poids de chaque critère de qualité, ce qui permet de calculer le score de chaque service atomique; le score est la somme pondérée des valeurs normalisée de la qualité de service:

$$Score\ (WS_{i,j}) = \sum_{i=1}^{Nb} (w_x * N_x) \quad (10)$$

Où:

Nb: représente le nombre de critères de qualité du service cible.

w_x: représente le poids d'un critère *x* de qualité de service qui est affectés par le client, afin de définir ses préférences par rapport à ce critère.

N_x : représente la valeur de qualité de service normalisée d'un critère *x*.

IV.6.2.Catégorisation

Le mécanisme de composition et classification crée deux tables qui contiennent les services sélectionnés, ces services sont classés sur la base des sous-requêtes (après sa décomposition) et les catégories d'existence. Dans la première table les services sont classés (chacun dans sa catégorie), sur la base de leur degré d'existence globale, et dans la deuxième table, les services sont classés en fonction de leurs scores QoS:

	Sous-requête				
Catégories	**Existence Complète**	**Existence Partielle**			**Existence Elémentaire**
	S_1	S_{n+1}	**Entrée**	**Sortie**	S_{k+1}
	S_2	S_{n+2}			S_{k+2}
	.	.	S_{m+1}	S_{r+1}	.
	.	.	.	.	.
	.	.	.	.	.
	.	.	.	.	.
	.	.	.	.	.
	.	.	.	.	.
	.	.	S_r	S_k	.
Services	S_n	S_m			S_z

Tableau IV.2 : Catégorisation de services basés sur la description textuelle du degré d'existence

Chaque colonne des deux tables représente une combinaison possibles des mots-clés de la requête; et chaque colonne contient deux autres colonnes : la première colonne contient les services de la catégorie « existence partielle (EP) » et la deuxième colonne contient les services de la catégorie « Existence Elémentaire (EE) » par rapport à la sous requête. Par exemple: supposons que nous avons 3 Mots-clés (MC) dans la requête du client, la ligne représentants les sous-requêtes s'affiche comme suit :

Sous-requête 1 **(MC_1, MC_2)**	**Sous-requête 2** **(MC_1, MC_3)**	**Sous-requête 3** **(MC_2, MC_3)**	**Sous-requête 4** **(MC_1)**	**Sous-requête 5** **(MC_2)**	**Sous-requête 6** **(MC_3)**
Partielle|Elementaire	Partielle|Elementaire	Partielle|Elementaire	Partielle|Elementaire	Partielle|Elementaire	Partielle|Elementaire

Tableau IV.3 : Catégorisation de services basés sur la sous-requête du client

IV.6.3.Tests de connectivité

Le mécanisme de composition et classification test la connectivité de chaque service sélectionné avec ces semblables, à partir des tables de catégorisations, et cela pour trouver toutes les combinaisons possibles entre eux, Ces tests sont menés essentiellement sur les

services qui appartiennent à des catégories de sous-requête différentes, afin de créer des services composites qui répondent aux exigences fonctionnelles et non fonctionnelle du client.

IV.6.3.1.Règles de connectivités

Les tests de connectivité entre services sont basés sur les règles suivantes:

1. Les services qui appartiennent à la catégorie « existence complète » sont exclus des tests de connectivité, puisque ils contiennent, dans leurs descriptions (arbres WSDL), tous les mots-clés de la requête du client.
2. Le mécanisme de composition et classification , commence toujours les tests de connectivité avec la catégorie de sous-requêtes qui contient le plus grand nombre de mots-clés ... Quand tous les services qui appartiennent à cette catégorie sont éliminés des tables de catégorisation, le mécanisme passe à la catégorie de sous-requêtes suivantes dans la table de catégorisation, et ainsi de suite.
3. Dans la même catégorie de sous-requêtes, le mécanisme de composition et classification, commence toujours les tests de connectivité avec le meilleur service qui appartient à la catégorie « existence partielle »... Quand tous les services de cette catégorie sont éliminés, le mécanisme utilise les services qui appartiennent à la catégorie « existence élémentaire » dans la même catégorie de sous-requêtes.
4. Dans la même catégorie «d'existence partielle ou élémentaire», le mécanisme commence toujours les tests de connectivité, par le service qui a le meilleur degré d'existence globale (dans la table organisée en fonction du degré d'existence) avec le service qui a le meilleur score de qualité; car le moteur de découverte et composition proposée, effectue l'opération de recherche de services, sur la base des exigences fonctionnelles et non -fonctionnelles des clients. Ce qui veut dire que le mécanisme de composition et sélection choisit les services alternativement dans les deux tables de la catégorisation afin de trouver un compromis entre ces deux aspects (aspects fonctionnel et non fonctionnel).
5. Dans la même catégorie « existence partielle » :
   - Lorsque le mécanisme de composition et classification, commence les tests de connectivité par un service qui appartient à la catégorie « existence partielle en entrée», le prochaine service choisi doit appartenir à la catégorie « existence partielle en sortie » dans la catégorie de sous-requête suivante (selon les $4^{ème}$ et $6^{ème}$ règles) et ainsi de suite ...et cela pour améliorer les chances d'avoir un lien de connectivité entre ces services.
   - Lorsque le mécanisme de composition et classification, commence les tests de connectivité avec le service qui appartient à la catégorie « existence partielle en sortie », le service prochain choisi doit appartenir à la catégorie « existence partielle en entré » dans la catégorie de sous-requête suivante (selon les $4^{ème}$ et $6^{ème}$ règles) et ainsi de suite ... et cela pour améliorer les chances d'avoir un lien de connectivité entre ces services.
6. Comme tous le mécanisme de découverte et composition, est entièrement basé sur les mots-clés de la requête, ce dernier, choisit toujours (lors du test d'une chaîne de connectivité), le service suivant, à partir de la catégorie de sous-requête suivante, qui contient les mots clés qui ne figurent pas dans les services qui composent la chaîne

de connectivité. L'objectif final est de créer un service composite qui contient un maximum de mots-clés de la requête.

7. Les tests de connectivité sont arrêtés, dans trois cas:
   - Lorsque, pour un service donné, tous les tests de connectivités possibles ont été effectué, puis le mécanisme choisit la meilleure chaîne de connectivité (voir l'étape de classification de service ci-dessous).
   - Lorsque la chaîne de connectivité contient tous les mots clés de la requête.
   - Quand il n'y a plus de services qui peuvent être composés.
8. Le mécanisme de composition et sélection élimine tout service des tables de catégorisation, s'il est accepté dans une chaîne de connectivité.

Le but de l'établissement de ces règles est de : réduire le nombre de tests de connectivité, améliorer les chances d'arriver à l'optimum global (le meilleur service composite possible), et réduire le temps de réponse.

IV.6.3.2.Distance entre arbres WSDL

Le mécanisme de composition et classification, utilise la distance d'édition entre sous arbres « portType » pour calculer le degré de connectivité, entre les services dont on veut tester la connectivité, afin de déterminer les meilleures chaines.

La distance d'édition entre arbres, est la méthode la plus efficace et la plus utilisé pour calculer la distance entre deux arbres, elle est basée sur trois opérations principales : (1) suppression des nœuds, (2) insertion de nœuds, et (3) ré-étiquetage de nœuds. L'idée consiste à trouver l'ensemble minimal d'opérations pour transformer un arbre en un autre.

Dans ce qui suit, nous présentons quelques définitions pour illustrer le processus qui permet de calculé le degré de connectivité entre deux services:

Définition 1 (connectivité entre services) : étant donné deux opérations $Opr_1 : In1_1, In1_2, In1_3, \dots, In1_n \longrightarrow Ot1_1, Ot1_2, Ot1_3, \dots, Ot1_m$, et $Opr_2 : In1_1, In2_2, In2_3, \dots, In2_r \longrightarrow Ot2_1, Ot2_2, Ot2_3, \dots, Ot2_z$, et soit $OTS = \{ Ot1_1, Ot1_2, Ot1_3, \dots, Ot1_m \}$ et $INS = \{ In1_1, In2_2, In2_3, \dots, In2_r \}$. Le degré de connectivité entre les Opr_1 et Opr_2 est le degré de similarité entre les arbres *OTS* et *INS*.

Définition 2 (Mapping entre arbres) : soit *InT* l'arbre des entrées d'un service, et InT[i] le nœud *i* de l'arbre InT, et soit OtT l'arbre des sorties et OtT[j] le nœud *j* de l'arbre OtT en traversant l'arbre en pré-ordre. Un Mapping entre les arbres InTOtT est un ensemble de M paires ordonnées (*i* , *j*), qui satisfont les conditions suivantes :
Pour tous (i_1, j_1); $(i_2, j_2) \in M$:

- $i_1 = i_2$ si et seulement si $j_1 = j_2$;
- InT[i_1] est sur la gauche de InT [i_2] i $\in$ OtT_2 [j_1] est sur la gauche de OtT [j_2];
- InT[i_1] est un ancêtre de InT [i_2] i $\in$ OtT2 [j_1] est un ancêtre de OtT [j_2]

Pour calculer la distance entre InT et OtT nous attribuons à chaque opération un coût, puis nous calculons les opérations minimales pour transformer InT à OtT.

Définition 3 (Le modèle de coût proposé) : nous proposons un nouveau modèle de coût pour la transformation d'un arbre à un autre, qui va avec la philosophie du mécanisme de

composition et classification proposée, cette philosophie qui consiste à utiliser la requête du client comme guide pour l'exécution de chaque étape du mécanisme. Nous proposons d'utiliser la similarité sémantique entre les mots-clés de la requête et les mots-clés des nœuds de chaque sous arbre « portType » pour calculer le coût d'édition:

$$Cost\ (delete\ (N)) = \left\{ \begin{array}{l} \dfrac{1 - sim\,(Q, N)}{Glob\ \_\ Exis\,(Q, InT) + Glob\ \_\ Exis\,(Q, OtT)} \text{ si } sim\,(Q, N) \neq 1 \\ \dfrac{1}{Glob\ \_\ Exis\,(Q, InT) + Glob\ \_\ Exis\,(Q, OtT)} \text{ si } sim\,(Q, N) = 1 \end{array} \right\} \quad (11)$$

$$Cost\ (insert\ (N)) = \left\{ \begin{array}{l} \dfrac{sim\,(Q, N)}{Glob\ \_\ Exis\,(Q, InT) + Glob\ \_\ Exis\,(Q, OtT)} \text{ si } sim\,(Q, N) \neq 1 \\ \dfrac{1}{Glob\ \_\ Exis\,(Q, InT) + Glob\ \_\ Exis\,(Q, OtT)} \text{ si } sim\,(Q, N) = 1 \end{array} \right\} \quad (12)$$

$$Cost\ (relabels\ (N_1, N_2)) = \frac{sim\,(Q, N_1) - sim\,(Q, N_2)}{Glob\ \_\ Exis\,(Q, InT) + Glob\ \_\ Exis\,(Q, OtT)} \quad (13)$$

Où:

Sim(Q, N): représente la similarité sémantique entre la requête du clientet le nœud de l'arbre cible.

Global_Exis (Q, InT) : représente le degré d'existence globale de la requête du client dans l'arbre cible des entrées.

Global_Exis (Q, OtT) : représente le degré d'existence globale de la requête du client dans l'arbre cible des sorties.

Explication 2 : le modèle du coût proposé est basé sur le gain ou la perte de similarité sémantique entre la requête du client et les nœuds: supprimés, insérés ou ré-étiquetés :

- La première formule représente le degré de perte sémantique de la requête par rapport au nœud ciblé, et cela relativement au degré d'existence globale de la requête dans les sous arbres qui représentent les entrées et sorties d'un service, et cela lorsqu'on supprime ce nœud.
- La deuxième formule décrit le fait que toute la sémantique de la requête est perdue par rapport au nœud ciblé, et cela relativement au degré d'existence globale de la requête dans les arbres qui représentent les entrées et sorties d'un service, et cela lorsqu'on supprime ce nœud.
- La troisième formule représente le degré de gain sémantique de la requête par rapport au nœud ciblé, et cela relativement au degré d'existence globale de la requête dans les sous arbres qui représentent les entrées et sorties d'un service, et cela lorsqu'on insère ce nœud.
- La quatrième formule décrit le fait que toute la sémantique de la requête est gagnée (ou préservée) par rapport au nœud ciblé, et cela relativement au degré d'existence globale de la requête dans les arbres qui représentent les entrées et sorties d'un service, et cela lorsqu'on insère ce nœud.
- Nous distinguons trois cas avec la Cinquième formule:
  1. *Sim(Q, N_1)-Sim(Q-N_2)>0*: représente le degré de gain sémantique de la requête par rapport au nœud *N_1*, et cela relativement au degré d'existence globale de la

requête dans les sous arbres qui représentent les entrées et sorties d'un service, et cela quand le nœud N_1 est ré-étiqueté.

2. $Sim(Q,N_1)-Sim(Q-N_2)< 0$: représente le degré de perte sémantique de la requête par rapport au nœud N_1, et cela relativement au degré d'existence globale de la requête dans les sous arbres qui représentent les entrées et sorties d'un service, et cela quand le nœud N_1 est ré-étiqueté.
3. $Sim(Q,N_1)-Sim(Q,N_2)=0$: cela signifie que N_1 reste inchangé; ce qui veut dire qu'il n'y a ni perte ni gain sémantique.

Définition 4 (gain sémantique d'une chaîne de connectivité) : le degré de connectivité d'une chaine de services est calculé en fonction de deux facteurs : (1) la somme des opérations qui permettent de transformer un sous arbre qui représente les entrées d'un service en un autre sous arbre qui représente les sorties du service correspondant dans la chaîne de connectivité, et (2) la somme des degrés de gain de similarité pour les opérations de transformation dans une chaîne de connectivité ,et cela pour choisir la meilleure chaîne entre toutes les chaînes de connectivité possibles pour un service donné. D'abord nous normalisons le degré de perte de similarité (lors de la suppression ou le ré-étiquetage des nœuds), ensuite nous calculons le degré de gain de similarité comme suit :

1. Normalisation :

$$Normalize\ (Cost\ (delete\ (N)) = 1 - Cost\ (delete\ (N)) \text{ pour un noued suprimé.} \quad (14)$$

$$Normalize\ (Cost\ (relable\ (N_1, N_2)) = 1 + Cost\ (relable\ (N_1, N_2)) \quad \text{Si } Sim(Q,N_1)-Sim(Q,N_2)<0 \quad (15)$$

2. Degré de gain de Similarité: le gain de similarité est calculé comme suit:

$$Gain\_Deg(conec\_chain) = \sum_{i=1}^{Nbc-1} (\sum_{j=0}^{NbD} NCD_j + \sum_{k=0}^{NbI} CI_k + \sum_{r=0}^{NbR} NCR_r) \quad (16)$$

Où:

Nbc: représente le nombre de services d'une chaîne de connectivité sous test.
NbD: représente le nombre de nœuds supprimés lors de la transformation d'un arbre d'entrées en un arbre de sorties dans une chaîne de connectivité sous test.
Nabi: représente le nombre de nœuds insérés lors de la transformation d'un arbre d'entrées en un arbre de sorties dans une chaîne de connectivité sous test.
NbR: représente le nombre de nœuds ré-étiquetés lors de la transformation d'un arbre d'entrées en un l'arbre de sortie dans une chaîne de connectivité sous test.
NCD: représente le coût normalisé pour une opération de suppression.
NCR: représente le coût normalisé pour une opération de ré-étiquetage.
CI : représente le coût de l'opération d'insertion.

3. Meilleure chaîne de connectivité : nous choisissons la meilleure chaîne de connectivité par l'application des règles suivantes :
   - La meilleure chaîne de connectivité est la chaîne de services qui utilise le moins d'opérations de transformations.
   - Si plusieurs chaînes de connectivité utilisent le même nombre d'opérations de transformations, la meilleure chaîne de connectivité est celle qui a le plus grand degré de gain de similarité.

IV.6.4. Classification des services

Le mécanisme de composition et classification, classe les services atomiques et composites en fonction de quatre facteurs : la catégorie d'existence, les attributs Cloud la qualité du service et le degré d'existence. L'algorithme proposé classe les services sur la base des règles suivantes :

1. La première classe ne contient que les services qui ont une « existence complète » de la requête du client dans les fichiers WSDL.
2. La deuxième classe ne contient que les services composites qui ont une « partielle existence » de la requête du client dans leurs fichiers WSDL.
3. La troisième classe ne contient que les services composites qui ont une « existence élémentaire » de la requête du client dans leurs fichiers WSDL.
4. Les services appartenant à la même classe sont classés selon le degré de similarité avec la partie de la requête du client, et qui décrit les caractéristiques Cloud des services souhaités. Pour calculer la similarité Cloud, on distingue trois cas:
   - Valeurs numériques: le score des valeurs numérique (NVS) est calculé en utilisant le modèle SWA présenté dans la section 6.1, parce que les caractéristiques numériques sont similaires aux caractéristiques non-fonctionnelles à augmenter. Tout d'abord, on normalise les valeurs numériques en utilisant la seconde formule de normalisation, puis nous calculons le NVS en utilisant les poids de chaque caractéristique numérique (choisie par le client afin de déterminer ses préférences par rapport à ces caractéristiques).
   - Valeurs textuelles: la similarité des valeurs textuelle (TVS) est calculée en utilisant la mesure de similarité sémantique présentée par [146] (section 5.3 ci-dessus) à base de WordNet.
   - Valeurs prédéterminées: dans ce cas, le client choisit la valeur qu'il souhaite directement à partir *d'une liste déroulante*, la valeur de similarité peut être 1 ou 0. La similarité entre la requête du client et les valeurs prédéterminée se calcule comme suit :

$$PVS = \frac{NCSV}{TNbEV} \tag{17}$$

Où:

PVS: représente la similarité entre la requête du client et les valeurs prédéterminées.
NCSV: représente le nombre de caractéristiques prédéterminées qui ont une valeur de similarité égale à 1.
TNbEV: représente le nombre total de caractéristiques prédéterminées que le client choisi.

Enfin, nous calculons la similarité Cloud, en utilisant la formule suivante:

$$CSim(Q, S_i) = \frac{\sum_{j=1}^{NbAS} \left(\frac{NVS_j + TVS_j + PVS_j}{3}\right)}{NbAS} \tag{18}$$

Où:

CSim: représente la similarité Cloud entre la requête du client et le service sélectionné S_i.

NBAS: représente le nombre de services atomiques qui entre dans la composition de S_i, si S_i est un service atomique, *NBAS* est égale à 1.

5. Si deux ou plusieurs services, qui appartiennent à la même classe, ont le même degré de similarité Cloud, nous utilisons les scores QoS pour classer ces services; on distingue deux cas:

   - Les services atomiques : nous utilisons la technique de pondération additive simple pour calculer le score de ce type de services.
   - Les services Web composites : lorsque nous évaluons la qualité d'un service composite, nous devons utiliser un ensemble de fonctions d'agrégations qui prennent en considération la manière de combinaison entre les services lors de la composition [147]. Nous distinguons quatre types de combinaisons : Séquentielle, parallèles, conditionnelles et boucle, le tableau suivant présente quelques formules qui servent à calculer le score QoS pour le temps de réponse et la fiabilité :

	Temps de réponse	**Fiabilité**
Séquentielle	$RT(CS) = \sum RT(AS_i)$	$R(CS) = \prod_{i=1}^{n} (e^{R(AS_i)})$
Parallèle	$RT(CS) = RT(AS_i) + Max(RT(AS_{i+1}),...,RT(AS_n)) + RT(AS_{n+1})$	$R(CS) = (e^{R(AS_i)}) + Min((e^{R(AS_{i+1})}),...,(e^{R(AS_n)}) + (e^{R(AS_{n+1})})$
Conditionnelle	$RT(CS) = \sum RT(AS_i)$	$R(CS) = \prod_{i=1}^{n} (e^{R(AS_i)})$
Boucle	$RT(CS) = n * \sum RT(AS_i)$	$R(CS) = (e^{R(AS_i)})^n$

Tableau IV.4: Exemple de calcul des scores QoS des services composites

Où:

CS: représente un service composite.

AS: représente le service atomique qui compose le *CS*.

6. Si plusieurs services qui appartiennent à la même classe, ont le même score QoS, on utilise le degré de l'existence globale pour les classer. Nous distinguons deux cas:

   - Les services web atomiques: le degré d'existence globale des services atomiques a été calculé à la fin de l'exécution du mécanisme de découverte et sélection.
   - Les services composites: le degré d'existence globale d'un service composite est la somme des degrés d'existence globale des services atomiques qui le composent divisé par le nombre de ces services:

$$Glob\ \_\ Exis\ \_\ Comp\ (CS_i) = \frac{\sum_{j=1}^{NbS} Glob\ \_\ Exis\ (AS_{i,j})}{NbS} \quad (19)$$

Où:

Glob_Exis_Comp(CS$_i$): représente le degré d'existence globale du service composite *i*.

Glob_Exis(AS$_{i,j}$): représente le degré d'existence globale du service atomique *j* qui entrent dans la composition du service *i*.

NbS : représente le nombre de services atomiques.

Algorithme 2 Mécanisme de composition et classification

Inpu1: (i) Requête du client ***Q*** (ii) Poids des attributs QoS et Cloud ***WC*** (iii) Arbres WSDL sélectionné ***SWT***

Output: (i) SaaS Composites (ii) SaaS classés

1:**Foreach *SWT*** sélectionné **do**

// Pré-classification de services de base QoS

2: **Forbach** Critère QoS **do**

3: Norm_QoS = Normalisation (*QoS_Critere*);

4: Score_QoS [id] = Som_Pondéré (Norm_QoS); //id: est l'identificateur du service sélectionné.

// Catégorisation

5: Création de deux tables de catégorisation comme expliqué dans la Section IV.6.2 ci-dessus;

// Tests de connectivites

6: **do**

7: **Foreach**Service appartenant à la catégorie d'existence partielle **do**

8: Prendre, de la table fonctionnelle, le service qui contient le plus grand nombre de mots clés et qui appartient à la catégorie existence partielle en entrée (*SWT* [id$_1$]);

9: Prendre, de la table non-fonctionnelle, le service qui contient le plus grand nombre de mots clés différents et qui appartiennent à la catégorie existence partielle en sortie (*SWT* [id$_2$]);

10: Dis_Edit_Arbre [PCC] = Dis_Edit_Arbre [PCC] +Calcul de la distance entre arbre *(SWT*[id$_1$], *SWT* [id$_2$]);

// PCC: est l'identificateur d'une Possible Chain de Connectivité.

11: Sauter au service suivant dans la table fonctionnelle ... et ainsi de suite;

12: **if** (n'importe quelle chaine de connectivité contient tous les mots clés) **then**

13: Arrêté les tests de connectivité;

14: Choisir la chaine actuelle;

15: Eliminer les services appartenant à la chaine de connectivité des deux tables;

16: **else** // après avoir testé toutes les chaines de connectivité possible

17: Sauter à la catégorie d'existence élémentaire;

18: Prendre, de la table fonctionnelle, le service qui contient le plus grand nombre de mots clés et qui appartient à la catégorie élémentaire *SWT* [id$_n$];

// n: est le rang du premier service choisi de la catégorie élémentaire.

19: Prendre, de la table non-fonctionnelle, le service qui contient le plus grand nombre de mots clés différents et qui appartiennent à la catégorie d'existence élémentaire *SWT*[id$_{n+1}$];

20: Dis_Edit_Arbre [PCC] = Dis_Edit_Arbre [PCC] + Calcul de la distance entre arbre *(SWT*[id_1], *SWT*[id_2])
21: Sauter au service suivant dans la table fonctionnelle ... et ainsi de suite;
22: **if** (n'importe quelle chaine de connectivité contient tous les mots clés)**then**
23: Arrêter les tests de connectivité;
24: Choisir la chaine actuelle;
25: Eliminer les services appartenant à la chaine de connectivité des deux tables;
26: **else**// après avoir testé toutes les chaines de connectivité possible
27: Choisir la meilleur chaine de connectivité avec le minimum d'opérations de transformation et le maximum de gain de similarité;
28: Eliminer les services appartenant à la chaine de connectivité des deux tables;
29: **if** (toutes les catégories partielles sont vides) **then**
30: **Foreach** Service appartenant à la catégorie d'existence Elémentaire **do**
31 Prendre, de la table fonctionnelle, le service qui contient le plus grand nombre de mots clés et qui appartiennent à la catégorie élémentaire *SWT*[id_1];
32: Prendre, de la table non-fonctionnelle, le service qui contient le plus grand nombre de mots clés différents et qui appartiennent à la catégorie d'existence élémentaire *SWT*[id_2];
33: Dis_Edit_Arbre [PCC] = Dis_Edit_Arbre [PCC] + Calcul de la distance entre arbre *(SWT*[id_1], *SWT*[id_2]);
34: Sauter au service suivant dans la table fonctionnelle ... et ainsi de suite;
35: **if** (n'importe quelle chaine de connectivité contient tous les mots clés) **then**
36: Arrêter les tests de connectivité;
37: Choisir la chaine actuelle;
38: Eliminer les services appartenant a la chaine de connectivité des deux tables ;
40: **else**// après avoir testé toutes les chaines de connectivité possible
41: Choisir la meilleur chaine de connectivité avec le minimum d'opérations de transformations et le maximum de gain de similarité;
42: Eliminer les services appartenant à la chaine de connectivité des deux tables;
43:**While**(Les tables fonctionnelles et non-fonctionnelles ne sont pas vides);
// Classification de services
44:**Foreach** SaaS atomique ou composite **do**
45: **if** (tous les mots clés de la requête existent dans le service WSDL file) **then**
46: Return: le service appartient à la première classe;
47: Calculer le degré de similarité Cloud selon la section 6.4;
48: Classer les services selon leurs similarité Cloud;
49: **if** (deux services ont la même similarité Cloud) **then**
50: Calculer les scores QoS des services composites selon la section 6.4;
51: Classer les services selon leurs scores QoS;
52: **if** (deux services ont le même score QoS) **then**
53 : Calculer le degré d'existence globale des services composites selon la section 6.4;
54: Classer les services selon leurs degré d'existence globale;
55: **if** (une partie des mots clés existe sur tous les sous arbres opérations … cas d'existence partielle) **then**
56: Return: le service appartient à la Seconde classe;
57: Exécuter les instructions: 47, 48, 49, 50, 51, 52, 53, et 54;
58: **if** (une partie des mots clés existe dans une partie des sous arbres … cas d'existence élémentaire)
59: Return: le service appartient à la troisième classe;

60: Exécuter les instructions: 47, 48, 49, 50, 51, 52, 53, et 54;

Algorithme IV.2 Mécanisme de composition et classification de services SaaS

IV.7. Complexité des algorithmes de découverte et composition

De façon générale, la complexité d'un algorithme sert à évaluer ses performances et offre une estimation de son efficacité en terme de temps d'exécution et d'espace mémoire.

Afin de pouvoir comparer entre plusieurs algorithmes qui traitent la même problématique (ex. mécanisme de découverte et composition) on calcule toujours la complexité dans les pires des cas avec une taille de données égale à *n*.

Dans notre cas on distingue deux types d'algorithmes :

- Algorithme de découverte et sélection: la complexité temporelle de l'algorithme de découverte et sélection est $O(n^3)$. Ceci peut être vérifié par une simple inspection de l'algorithme. Mais le fait que la complexité est d'ordre n^3 ne signifie pas que l'algorithme sera lent, parce qu'il est exécuté sur des nœuds Cloud menu de puissantes capacités de calcul et stockage (chapitre 5).
- Algorithme de composition et classification: la complexité temporelle de l'algorithme de composition et classification est $O(n^2)$. Ceci peut être vérifié par une simple inspection de l'algorithme.

IV.7. Conclusion

La contribution présenté dans ce chapitre est la spécification d'une approche de : découverte, sélection, composition, et classification de services SaaS.

Ainsi le prochain chapitre présente notre contribution sur le plan architectural pour la découverte, sélection et composition de Software-as-a-Service. Nous nous sommes également inspirés des architectures et modèles de gestion de ressources basées agent dans le Cloud pour proposer une architecture complète qui s'occupe de l'exécution des algorithmes de matching présentés dans ce chapitre.

V

Contribution 2: Architecture du système

V.1. Introduction

La croissance exponentielle en nombre et possibilités, qu'offrent les architectures orientées services (SOA) dans le contexte de la technologie Cloud, essentiellement pour les services Internet, nous pousse à élaborer une nouvelle architecture basée Cloud pour l'automatisation de la découverte et composition de services SaaS.

Contrairement aux moteurs Google ou Yahoo, qui sont consacrés à la recherche de tous les types d'informations, nous proposons un nouveau moteur conçu uniquement à la découverte et composition de services SaaS, et qui s'occupe de l'exécution des algorithmes présentés dans le chapitre précèdent.

Ce chapitre est organisé comme suit :
Dans la section 2 nous présentons les objectifs globaux et techniques de l'architecture proposée. Dans la section 3 nous présentons les aspects généraux du système proposé. Dans la section 4, nous présentons en détail les composants de l'architecture proposée. Dans la section 5, nous présentons le mode de fonctionnement de chaque composant. Dans la section 6, nous présentons les principaux comportements d'agents, ainsi que les différents algorithmes de gestion de ressources Cloud. Dans la section 7, nous décrivons les bénéfices de la combinaison entre agent et Cloud. Enfin, la section 8 est la conclusion de ce travail.

V.2. Objectifs techniques du système

Les principales exigences prises en considération lors du développement du système proposé sont les suivantes :

- Exécution du mécanisme de découverte et composition : le mécanisme de découverte et composition de SaaS est l'élément clé pour le rendement du moteur proposé, car c'est ce mécanisme qui assure le niveau de qualité des résultats rendus aux clients. C'est pour cette raison que le premier objectif du moteur proposé consiste, à fournir tous les moyens virtuels et logiques dont le système dispose, afin d'assurer un déploiement efficace de tous les modules du mécanisme de découverte et composition.
- Garanti de l'évolutivité massive aux ressources Cloud : le client ne doit pas se soucier des détails du plan d'approvisionnement de ressources, le moteur doit recevoir les requêtes des clients, et déterminer les ressources nécessaires afin de les traiter, et cela sans aucune intervention des clients.
- Gestion des ressources internes: le moteur de découverte et composition de services, doit assurer, la gestion des machines virtuelles, des CPUs, des disques durs, et des instances des différentes applications Cloud. Le système de gestion de ces derniers, doit être caché aux utilisateurs mais en assurant un certain niveau de qualité, notamment en ce qui concerne le temps de réponse.

- Gestion des ressources externes: le moteur de découverte et composition de services doit également gérer, l'enregistrement des nœuds externe au système, la distance entre ces nœuds et les nœuds principaux, le temps de déploiement des applications Cloud, et le nombre de nœuds visités lors de la recherche de nouvelle ressources externe. Le système de gestion des ressources externe, doit être, également, caché aux clients.
- Modélisation du système de gestion avec des SMA, afin d'avoir une gestion Intelligente des ressources Cloud. Les modules qui composent le moteur proposé doivent être modélisés avec des agents, de telle sorte que chaque tâche principale doit être supervisé et exécuté par un agent spécifique. Outre, les agents doivent coopérer et collaborer, et leurs comportements doivent être harmonieux, afin d'éviter des conflits éventuels lors de la gestion des ressources virtuelles et logiques.
- Gestion de la liaison entre les parties du système : la liaison entre les principaux nœuds du moteur proposé, doit être géré par un système mobile et autonome, et cela vu les risques et défis rencontrés dans les systèmes Cloud computing. Les agents mobiles sont la meilleure alternative pour répondre à ces défis, car ce type d'agents est autonome et peut s'adapter aux différentes situations et risques, surtout, pour des situations rencontrées sur des environnements instables et incertains comme Internet, en plus, ce type d'agent et célèbre pour ces capacités à transmettre l'information en toute sécurité.

Le premier objectif est le plus important, car il s'agit d'implémenter et d'exécuter les différents modules du mécanisme de découverte et composition, et de répondre aux exigences fonctionnelles et non-fonctionnelles du client. Autrement dit, le mécanisme de découverte et composition, est la relation directe qui lie le client au moteur proposé.

Les deuxième, troisième et quatrième objectifs, sont essentiels pour la gestion des ressources Cloud. Suivre ces objectifs, permet de créer un système de supervision et de contrôle de ressources draconien, qui permet aux clients, de ne plus se soucier du système sous-jacent, et qui gère les infrastructures sur lesquelles sont déployés les différents nœuds Cloud.

Les deux derniers objectifs, sont importants pour la combinaison entre deux technologies futures, à savoir, technologie Cloud et technologie d'agents. Cette combinaison permet de créer une nouvelle génération de services orientés agents, qui permet de pousser les limites et les possibilités de gestion de ressources, vers des systèmes de gestion plus intelligents et plus adaptés aux besoins des clients et aux changements et développement rapide d'Internet.

V.3. Présentation du système

La conception de notre moteur de découverte et composition, est conforme avec les exigences techniques du mécanisme de matching définies dans le chapitre précèdent. Les principaux nœuds du système proposé peuvent être situés sur différents emplacements géographiques, comme ils peuvent être regroupés sur un seul Datacenter. Chaque nœud a son propre Infrastructure-as-a-Service, gérée par un système multi agents, qui a son catalogue d'information, sur lequel sont regroupés les paramètres descriptifs des capacités des ressources, ainsi que les prix. En plus, ce catalogue contient toutes les informations sur les moyens matériels et logiciel dont disposent les autres Nœuds du système.

Les ressources des principaux nœuds, sont des machines virtuelles, des disques de stockage, des CPUs, et des nœuds externes qui représentent des IaaS pour le déploiement d'applications Cloud.

On conçoit également notre système, de telle façon à ce que chaque nœud dispose de son propre système de gestion de ressources. Ce qui signifie que chaque nœud est capable de déployer ces services et applications par l'intermédiaire de son propre API.

De façon générale, le moteur proposé est responsable de l'approvisionnement de ressources, cela signifie que le système de gestion de ressources de chaque nœud décide quand, sur quel nœud et dans quel contexte les composants du moteur seront déployés.

Au cœur du moteur proposé, le système de gestion de ressources, reçoit les requêtes des clients, évalue les capacités virtuelles et logiques du système, et éventuellement construit un plans d'approvisionnement de ressources et distribue les taches sur les différents agents, pour lancer, enfin l'exécution du mécanisme de découverte et composition.

Enfin chaque nœud du moteur proposé, gère le déploiement de ses propre applications selon ses besoins en ressources de ses composants, par conséquent l'Independence inter-nœud est préservée, ce qui permet de créer un système hautement tolérant aux fautes, et permet de réduire le rôle du gestionnaire de ressources (le rôle sera de simplement copier les codes nécessaires, pour l'implémentation des agents de gestion de ressources sur les nœuds de déploiement).

V.4. Composants du système

Dans cette section nous présentons, les composants du moteur de découverte et composition proposé:

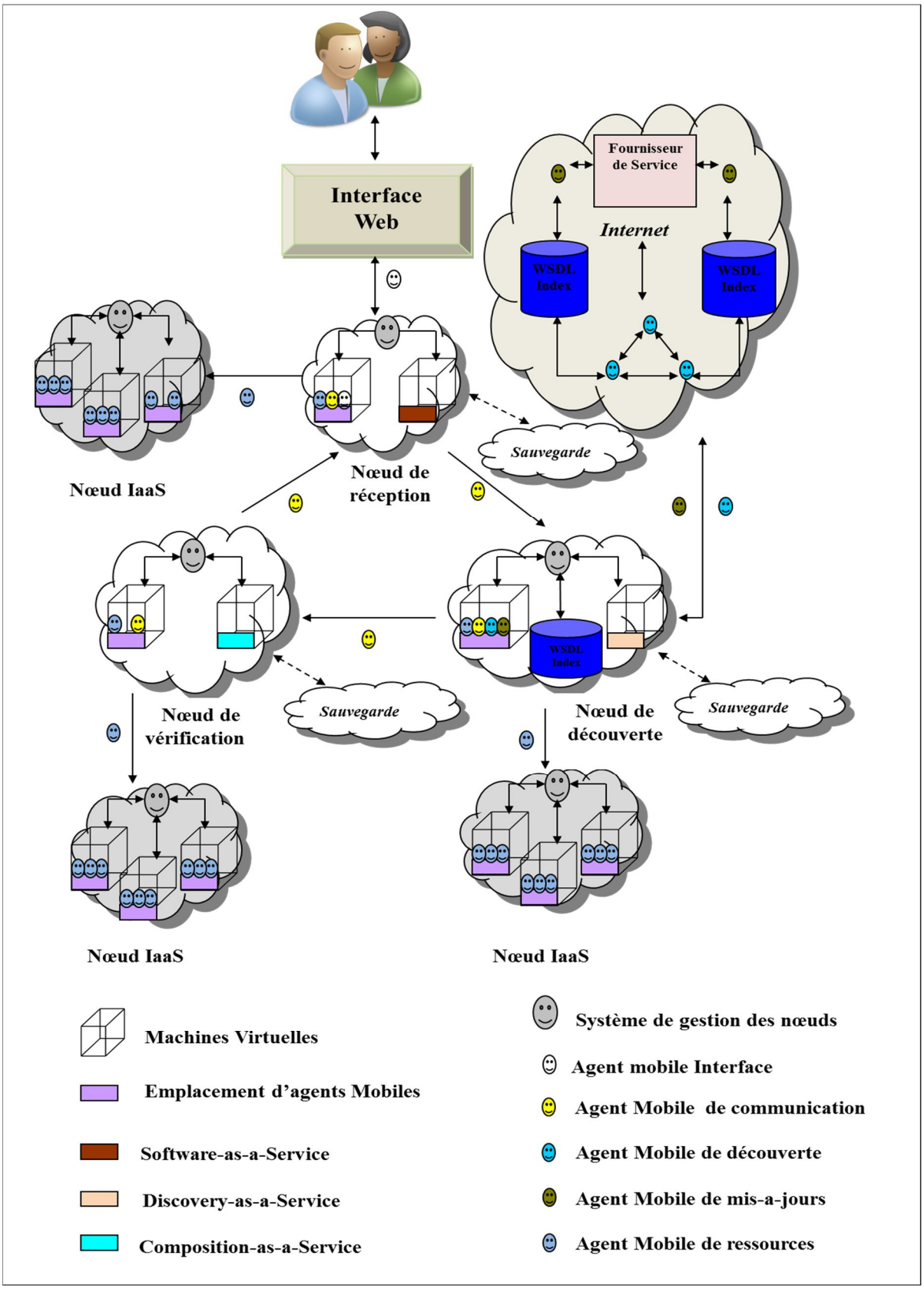

Figure V.1: Architecture globale du système proposé.

V.4.1.Interface du système

L'interface permet aux clients d'introduire:

- La Requêtes sous forme textuelle: pour chercher les services qui répondent à leurs exigences fonctionnelles(en langage naturel)
- Les poids des paramètres QoS: afin que les clients puissent exprimer l'importance de chaque paramètre QoS, selon leurs besoins. L'Interface leurs permet d'introduire le poids associé à chaque paramètre (*ex.* temps de réponse, disponibilité, coût …etc.).
- Les poids des paramètres Cloud.
- Les valeurs Cloud prédéterminées.

L'interface offre, également, aux clients:

- Une page web "Contacter nous": qui offre aux clients la possibilité, de contacter le service d'assistance de notre moteur de découverte et composition, et cela en cas de défaillance du système ou si le client a une interrogation ou réclamation spécifique.
- Un espace pour l'affichage des résultats.

V.4.2. Les Composants Cloud

Cette partie est composée de trois principaux nœuds Cloud; chaque nœud dispose de ses propres capacités de stockage et de traitement. La partie Cloud est également gérée par un système multi-agents spécifique à ses objectifs:

V.4.2.1. Nœud de réception

Le Nœud de réception a pour principaux objectifs: la réception des requêtes des clients, et la création d'un fichier pour stocker les exigences fonctionnelles et non fonctionnelles de chaque requête, ce nœud est basé sur trois modèles de livraisons de services Cloud:

1. Software-as-a-Service (SaaS) pour: la codification des requêtes des clients afin de mieux les identifier, et la gestion des problèmes d'ordre sécuritaire
2. Storage-as-a-Service (StaaS) : pour stocker les requêtes des clients et leurs identifiants, stocker les services virtuels composites retournés par le nœud de vérification et classification, et stocker les différents codes (codes sources) relatifs au fonctionnement de ce nœud
3. Human-Support-as-a-Service (HsaaS) [148]: Cette partie traite les exigences particulières des clients : par exemple, si le client ne reçoit pas de résultats convenables, il peut envoyer des suggestions ou des réclamations aux gérants du système, pour que l'administrateur humain optimise les services offerts par le système proposé, et cela en prenant en considération les remarques des clients. L'autre fonction de cette partie, est la mise en place d'un SLA, entre les principaux nœuds du système (Nœud de réception, Nœud de découverte, Nœud de vérification classification) et les fournisseurs de services Cloud IaaS, pour assurer un déploiement rapide et efficace des composants de chaque nœud.

En plus, le nœud de réception a comme support, un nœud de sauvegarde (backup en anglais), qui stocke une copie de toutes les informations stockées sur le nœud de réception (requête, identifiant, services virtuels composites, et les codes sources), afin d'assurer

l'intégrité du système en cas d'échec. Le nœud de sauvegarde, est caractérisé par: (i) il est utilisé temporairement par le système de gestion de ressources, en cas ou le nœud de réception est dans un état d'échec, et cela pour offrir assez de temps aux agents mobiles ressources (Section 4.4.5), de déployer les services offerts par le nœud de réception sur le nœud IaaS ciblé, (ii) le nœud de sauvegarde assure: la disponibilité et l'instantanéité des services offerts par le nœud de réception.

V.4.2.2. Nœud de Découverte

Le nœud de découverte héberge le mécanisme de découverte et sélection de services, vue dans le chapitre précèdent, ce nœud est basé sur trois modèles de livraison de services Cloud :

1. Discovery-as-a-Service (DiaaS): c'est une variante de SaaS, qui exécute le mécanisme de découverte et sélection de service, sur la base des fichiers clients reçus du nœud de réception et les fichiers WSDL stockés dans l'index.
2. Storage-as-a Service: pour stocker: les fichiers clients et les fichiers WSDL sur l'index, afin de les rendre proche du DiaaS pour réduire le temps d'accès.
3. Developper-as-a-Service (DeaaS): il s'agit d'une combinaison entre PaaS (Platform-as-a-Service) et HsaaS. Lorsqu'un client envoie des suggestions spécifiques concernant le système de découverte, c'est aux développeurs (les développeurs sont des ingénieurs qui maintiennent et optimisent le nœud de découverte), d'optimiser le mécanisme de découverte et sélection, et cela en utilisant des outils de développement issu du PaaS.

Enfin comme pour le nœud de sauvegarde du nœud réception, celui du nœud de découverte sert à stocker les données et les codes de ce dernier.

V.4.2.3. Nœud de Vérification et Classification

Le nœud de vérification et classification héberge le mécanisme de composition et classification de service. Il est composé de trois modèles de livraison de service Cloud:

1. Composition-as-a-Service (CaaS): est une variété de SaaS, responsable de la création des services virtuels composites. Cette création se fait sur la base des fichiers clients reçus du nœud de réception et des services atomiques sélectionnés reçus du nœud de découverte, et cela en exécutant le mécanisme de composition et classification de services.
2. Storage-as-a-Service: pour stocker : les fichiers clients, les services atomiques sélectionnés, les scores QoS, les scores Cloud, les chaînes de connectivité (temporairement), et les services virtuels composites (le temps de les transmettre au nœud de réception).
3. Developer-as-a-Service : Lorsqu'un client envoie des suggestions spécifiques concernant le système de composition, c'est aux développeurs d'optimiser le mécanisme de composition et classification, en utilisant des outils de développement issus du PaaS.

Enfin comme pour le nœud de sauvegarde du nœud de découverte, celui du nœud de vérification et classification sert à stocker les données et les codes de ce dernier.

V.4.2.4. Les Nœuds Infrastructure-as- a-Service

Pour assurer une évolutivité massive des ressources de chaque nœud du moteur proposé, nous utilisons des nœuds qui offrent des services du type Infrastructure-as-a-Services. Parce que, le rôle principal d'une infrastructure Cloud est l'hébergement et le déploiement de:

- Tout type de Software-as-a-Service : dans le cas du moteur proposé il s'agit du : DiaaS et CaaS.
- Tout système de storage-as-a-Service : dans le cas du moteur proposé il s'agit de stocker les fichiers des clients, fichiers WSDL et les services virtuels composites.
- Tout système du genre Humain-Support-as-a-Service : dans le cas du moteur proposé il s'agit du : HsaaS et DeaaS.

Quand n'importe quel nœud a besoin d'augmenter les capacités de ces ressources, son système de gestion de ressources consulte le catalogue des services IaaS disponibles, pour choisir le plus adéquat d'entre eux, et il crée un agent mobile ressource pour transmettre les données et les codes nécessaire aux nœuds Infrastructure-as-a-Service, et cela afin de déployer ces services.

V.4.3. Systèmes de gestion des nœuds

Chaque nœud a son propre système de gestion, qui est composé d'un système multi-agent:

V.4.3.1. Système de gestion du nœud de réception

Le système de gestion du nœud de réception est composé de trois principaux agents: Agent administrateur, Agent d'infrastructure, et Agent statistique :

- Agent administrateur (AA) : les fonctions de cet agent sont les suivantes: (i) l'initiation du composant SaaS et l'agent statistique (ii) la création des fichiers clients, (iii) Gestion du composant StaaS , (iv) Création des agents mobiles , (vi) Mise-à-jour du nœud de sauvegarde , (vii) La sélection et la composition de ressources Cloud supplémentaires sur les nœuds IaaS ; (viii) initiation de l'agent d'infrastructure , (ix) La négociation et l'élaboration des SLA avec les fournisseurs de services IaaS , (x) l'orchestration de l'accès et l'utilisation du nœud de sauvegardes et des Ressources internes de chaque nœuds , (xi) la mise-à-jour des informations stockées sur la Table d'Auto-Approvisionnement de Ressources (TAAR) (section 4.5).
- Agent d'infrastructure (AIf) : les objectifs de cet agent sont : (i) La gestion des machines virtuelles (MV) (création , destruction , extension ou réduction des capacités) , (ii) l'orchestration de l'accès et l'utilisation des machines virtuelles , (iii)distribution des tâches sur les machines virtuelles , (iv) la sélection et la composition de ressources supplémentaires au sein de chaque nœud principal (réception, découverte ou nœud de vérification et classification) , (v) La supervision des ressources physiques (une ressource peut être disponible, occupé ou en panne) , (vi) remplacement des ressources virtuelles en état d'échec ,(vii) prévoir les besoins futurs de chaque nœuds basés sur le taux d'utilisation des ressources , (viii) Tester l'intégrité et le fonctionnement des ressources des nœuds IaaS.
- Agent statistique (AS) : les rôles de cet agent sont :

(i) La supervision de l'opération d'indexation des questions et des réclamations des clients en trois catégories :

1. Catégorie Administrative : lorsque la réclamation du client est une question d'ordre générale, ou sur le contrat SLA, elle est envoyée au composant HsaaS.
2. Catégorie Découverte : lorsque la réclamation du client est sur la qualité des services atomiques, la demande est envoyée au composant DeaaS dans le nœud de découverte.
3. Catégorie Composition: lorsque la réclamation du client est sur la qualité des services virtuels composites, elle est envoyée au DeaaS dans le nœud de vérification et classification.

(ii) La création d'agent mobile de communication pour transmettre les questions et les réclamations des clients.

V.4.3.2. Système de gestion du nœud de Découverte

Le système de gestion du nœud de découverte est composé de quatre principaux agents: Agent Administrateur, Agent de prétraitement, Agent de matching et Agent d'infrastructure:

- Agent administrateur: cet agent assure les mêmes fonctionnalités que l'AA du nœud de réception (à partir de (iii)à(xi)), en plus il assure les fonctionnalités suivantes: (i) Recevoir les fichiers des clients, (ii) initiation de l'agent de prétraitement , (iii) Mise-à-jours de WordNet.
- Agent de prétraitement (AP) : cet agent a pour fonctions:(i) l'initiation du DiaaS et de l'agent matching, (ii) superviser l'opération de la création des arbres WSDL, (iii) superviser l'opération du prétraitement des requêtes.
- Agent Matching (AM) : les fonctions de cet agent sont : (i) initiation du DiaaS, (ii) superviser l'opération de sélection des services atomiques, (iii) superviser l'opération de la préparation à la composition.
- Agent d'infrastructure: cet agent assure les mêmes fonctionnalités que l'Agent d'infrastructure du nœud de réception.

V.4.3.3. Système de gestion du nœud de vérification et classification

Le système de gestion du nœud de vérification et classification est composé de cinq type d'agents : Agent administrateur, Agent Qualité, Agent catégorisation, Agent composition, et Agent d'infrastructure.

- Agent administrateur : Cet agent assure les mêmes fonctionnalités que l'AA du nœud de réception (à partir de (iii) à (xi)), en plus il offre les fonctionnalités suivantes : (i) réception des fichiers clients et des services sélectionnés, (ii) Initiation de l'agent qualité.
- Agent qualité (AQ) : les objectifs de cet agent sont : (i) Initiation du composant CaaS et de l'agent catégorisation, (ii) il supervise l'opération de calcul des scores QoS des services atomiques, (iii) il supervise l'opération de calcul des scores QoS des services composites, (iv) il supervision de l'opération de pré-classification de services.

- Agent catégorisation (ACa):les fonctionnalités de cet agent sont les suivantes: (i) initiation du composant CaaS et de l'agent composition, (ii) il supervise l'opération de division de la requête en un ensemble de sous-requêtes, (iii) il supervise l'opération de création des tables de catégorisation.
- Agent composition (AC): les rôles de cet agent sont les suivants: (i) initiation du composant CaaS, (ii) il supervise les tests de connectivité entre les services sélectionnés, (iii) il supervise l'opération de classification des services atomiques et composites.
- Agent d'infrastructure: Cet agent assure les mêmes fonctionnalités que l'Agent d'infrastructure du nœud de réception.

V.4.3.4. Système de gestion des nœuds infrastructures

Quand un nœud à besoins des ressources supplémentaires, l'agent mobile ressources transmet les données et les codes nécessaires au déploiement : des composante Cloud (*ex.* Discovery-as-a-Service, Composition-as-a-Service ... etc.), et du Système de gestion des ressources du nœud qui a besoin de ressources supplémentaires. La transmission se fait vers le nœud IaaS cible, afin d'exécuter le code qui permet de déployer les composants, et les différents agents du système de gestion. De cette façon le nœud IaaS devient une copie du nœud déployé ce qui lui permet de fournir les mêmes services que ce dernier.

V.4.4.Agents Mobile

Afin d'avoir un système de communications à la fois intelligentes et efficaces pour la liaison entre les différents composants du système proposé nous utilisons un ensemble d'agents mobiles:

V.4.4.1. Agent Mobile Interface (AMI)

L'agent Interface (IA) est responsable de: (i) transmettre la requête et les réclamations des clients aux nœuds de réception, (ii) affichez les services virtuels composites aux clients. Cet agent est composé de: module de traitement, module de communication avec l'agent administrateur du nœud de réception (MCAA), module de communication avec le client (MCC) et une base de données pour stocker la requête du client et les services composites.

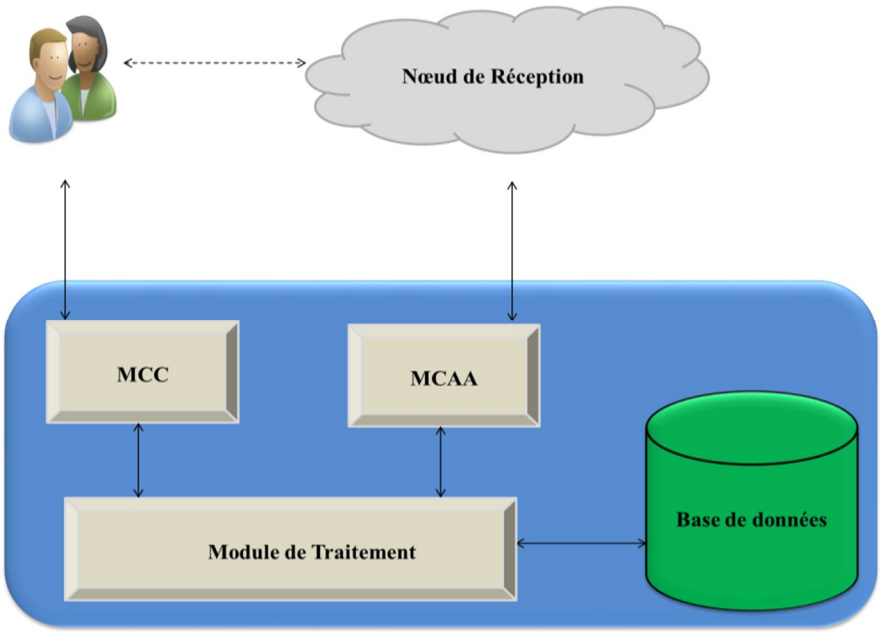

Figure V.2: Architecture de l'agent mobile interface.

V.4.4.2. Agent Mobile Communication (AMC)

Les fonctions de l'agent communication (RMR) sont: (i) la transmission des fichiers des clients et des réclamations aux nœuds de découverte et de vérification, (ii) la transmission des services sélectionnés à partir du nœud de découverte au nœud de vérification, (iii) Transmettre les services virtuel composites vers le nœud de réception (iv) Transmettre les informations d'TAAR entre les nœuds SDCE-ACS. Cet agent est composé de: module de traitement, module de communication avec les systèmes de gestion de ressources (MCSG) et une base de stockage.

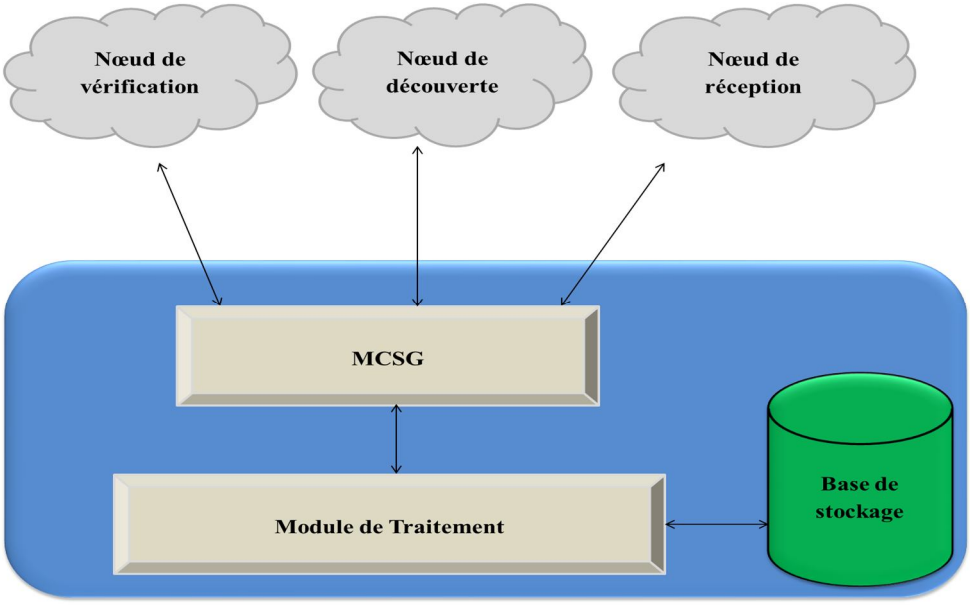

Figure V.3: Architecture de l'agent mobile de communication.

V.4.4.3. Agent Mobile Découverte (AMD)

Le but de l'agent de découverte est la recherche des services qui ne sont pas indexés dans l'Index WSDL du nœud de découverte. Quand un nouveau service est trouvé par l'agent de découverte, il apporte son fichier WSDL au nœud de découverte pour qu'il soit traité et stocké dans l'index WSDL. Pendant chaque période de temps, l'agent administrateur crée et lance un certain nombre d'agents mobiles de découverte (ce nombre est égal au nombre des régions Internet; chaque région Internet regroupe les fournisseurs et publicateurs de services, qui sont proches les uns des autres), pour la recherche de nouveaux services. Cet agent est composé de: module de traitement, module de communication avec l'agent administrateur du nœud de découverte (MCAA), module de communication avec d'autre agent découverte (MCAD) pour éviter la recherche dans les mêmes régions Internet, module de communication avec fournisseurs de services (MCFS), et une base de stockage pour stocker les fichiers WSDL et les paramètres QoS des services trouvés.

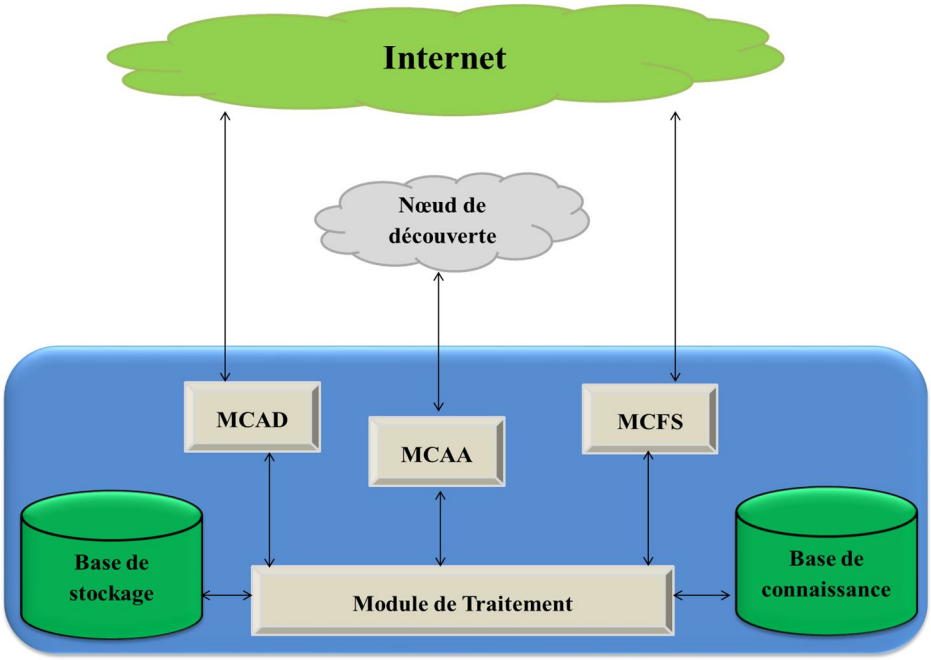

Figure V.4: Architecture de l'agent mobile de découverte.

V.4.4.4. Agent mobile de Mise-à-jours(AMM)

L'un des problèmes les plus importants de l'utilisation des services est le changement de la qualité du service sollicité par les clients. D'autre part, un fournisseur de services Internet peut changer les fonctionnalités offertes par ses services en ajoutant ou en supprimant un certain nombre de fonctionnalités. Le but de l'agent mise-à-jour consiste à vérifier, pendant chaque période de temps, les changements, dans la qualité ou les fonctionnalités des services enregistrés sur l'index WSDL du nœud de découverte. L'agent de mise-à-jour, encapsule dans sa «base de données un certain nombre de fichiers WSDL indexées dans le nœud de découverte et leurs valeurs de qualité de service. L'agent migre ensuite vers les fournisseurs de services Internet pour vérifier s'il y a des changements. Si l'agent de mise-à-jour détecte des changements, il actualise l'index WSDL. Cet agent est composé de: module de traitement pour comparer entre les fichiers WSDL indexés et les fichiers WSDL publiés, module de communication avec l'agent administrateur du nœud de découverte (MCAA), module de

communication avec fournisseurs de services (MCFS), une base de stockage pour stocker les fichiers WSDL et leur valeurs de qualité de service ainsi que les paramètres Cloud.

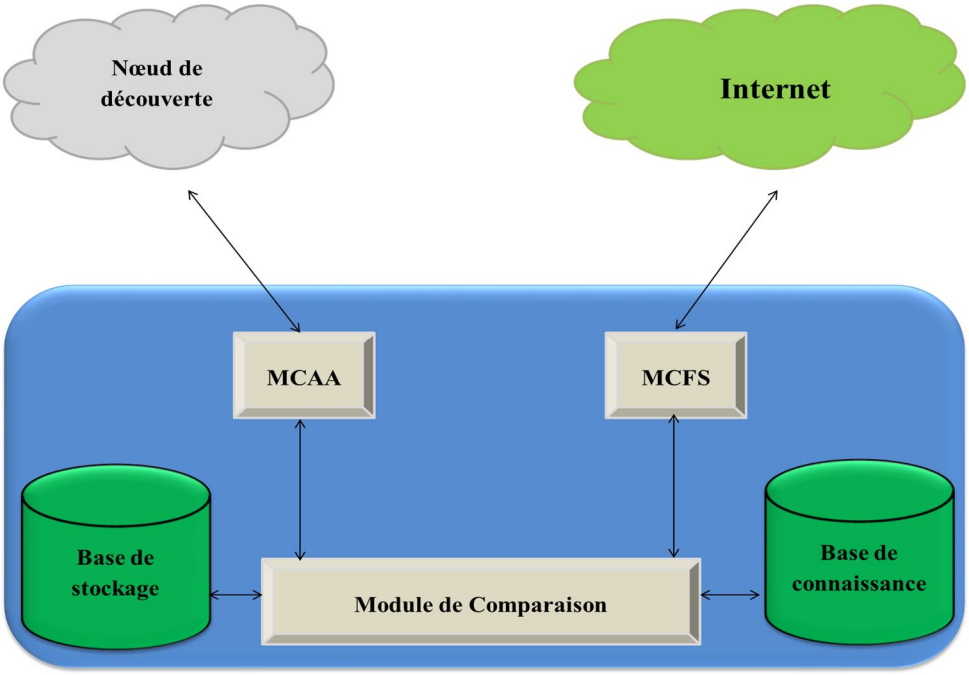

Figure V.5: Architecture de l'agent mobile de mis-a-jours.

V.4.4.5. Agents mobiles ressources (AMR)

Comme nous l'avons mentionné dans les sections précédents, l'agent ressources est responsable de la recherche de nouvelles ressources Cloud, afin d'assurer l'évolutivité massive des ressources du moteur proposé. L'agent administrateur, crée et lance, pendant chaque période de temps, un ensemble d'agents ressources sur Internet (le nombre des agents ressources est égal au nombre des régions Internet), afin de chercher les fournisseurs de services Cloud qui offrent une Infrastructure-as-a-Service, ensuite c'est à l'agent administrateur d'analyser la qualité de l'infrastructure trouvée par l'agent ressources .Si l'AA décide que l'IaaS est approprié au déploiement des différentes composantes Cloud (Storage-as-a-Service, Discovery-as-a-Service ... etc.), il envoie un message à l'administrateur humain, afin qu'il puise lancer la négociation avec le fournisseur de services d'infrastructure, sur les termes du contrat de déploiement des systèmes de gestion et des composants Cloud ,et cela en cas de besoin.

L'agent ressources est composé de: modules de traitements, module de communication avec les agents administrateurs de chaque nœud (MCAA), d'une base de données(les valeurs de qualité des services d'infrastructure trouvées), d'un module de communication avec les autres agents ressources (MCAR) et cela pour éviter de chercher sur les mêmes régions de Internet, et d'un module de communication avec les fournisseurs de services d'infrastructure Cloud (MCFSIC).

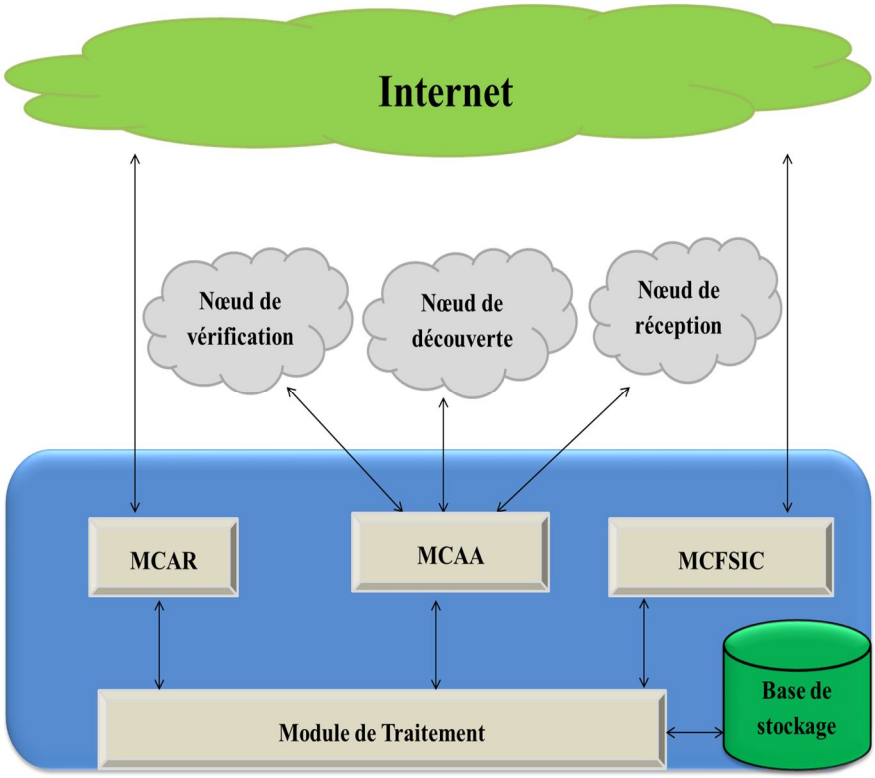

Figure V.6: Architecture de l'agent mobile de ressources.

V.4.5.Table d'Auto-Approvisionnement de Ressources (TAAR)

Chaque nœud du moteur proposé a son propre TAAR. Les TAAR sont utilisés et mis-à-jours par les agents AAs et AIfs, afin d'enregistrer et consulter les informations, sur les capacités et les types de ressources (Stockage, Traitement, et Networking) offertes par chaque nœuds principales, ainsi que sur les fournisseurs de service Cloud IaaS. Lorsque l'agent mobile ressources trouve un nouveau fournisseur de services IaaS , il envoie l'AA du nœud cible : (i) Le type des ressources IaaS trouvées , (ii) Les capacités de stockage , (iii) le nombre de serveurs , (iv) la capacité de la bande passante , v) le type d'outils de déploiement automatique , (vi) les coûts de déploiement et utilisation , (vii) la position géographique des ressources (pour calculer la distance entre les principaux nœuds, entre ces derniers et les bases des clients qui sont dans des heures de pointe), (viii) Les pré-conditions de l'utilisation des ressources IaaS , (ix) l'adresse du fournisseur de services IaaS , (x) le nombre de processeurs et unités de déploiement sur le réseaux. Lorsque l'AA, de tout nœud, reçoit ces informations, il met à jour son TAAR,et crée un agent mobile de communication pour l' envoyer à tous les nœuds du système. La TAAR permet aux AAs ainsi qu'aux AIfs de vérifier l'état des ressources sur le Cloud, à l'intérieur comme à l'extérieur des principaux nœuds, et cela afin de coordonner leurs «comportements »afin d'obtenir plus d'intelligence et d'efficacité pour la gestion des ressources virtuelles et logiques sur le Cloud.

V.5. Processus de découverte et composition

Dans cette section nous vérrons les processus complète d'exécution des méchanismes de découverte et compositon de services SaaS.

V.5.1. Fonctionnement des nœuds

Dans ce qui suit nous détaillon les étapes de fonctionnement de chaque nœud.

V.5.1.1. Fonctionnement général du système

Le scénario général de l'utilisation du moteur proposé est décrit par les étapes suivantes:

1. Le client accède à l'interface Web pour saisir sa requête ou sa réclamation.
2. L'agent d'interface amène la requête ou la réclamation au nœud de réception.
3. Le système de gestion du nœud de réception (SGNR) crée et stocke le fichier du client.
4. Si c'est une requête, le système de gestion du nœud de réception, crée deux agents de communication afin, de les envoyer (avec le fichier client) aux nœuds de découverte et de vérification (Si c'est une réclamation, le système de gestion index celle-ci, et l'envoie au nœud approprié en utilisant un agent de communication).
5. Le système de gestion du nœud de découverte (SGND) et du nœud de vérification (SGNV) reçoit et stocke le fichier ou la réclamation du client.
6. Le système de gestion du nœud de découverte, lance l'exécution du mécanisme de découverte et sélection pour choisir les meilleurs services qui correspondent à la requête du client.
7. Le système de gestion du nœud de découverte, crée un agent de communication pour envoyer les services choisis au nœud de vérification et classification.
8. Le système de gestion du nœud de vérification et classification reçoit et stocke les services sélectionnées.
9. Le système de gestion du nœud de vérification et classification, lance l'exécution du mécanisme de composition et classification de service, afin de crée un ensemble de services virtuels composites qui correspondent à la requête du client.
10. Le système de gestion du nœud de vérification et classification crée un agent de communication pour envoyer les résultats vers le nœud de réception.
11. Le système de gestion du nœud de réception crée un agent Interface pour afficher les services virtuels composites au client.

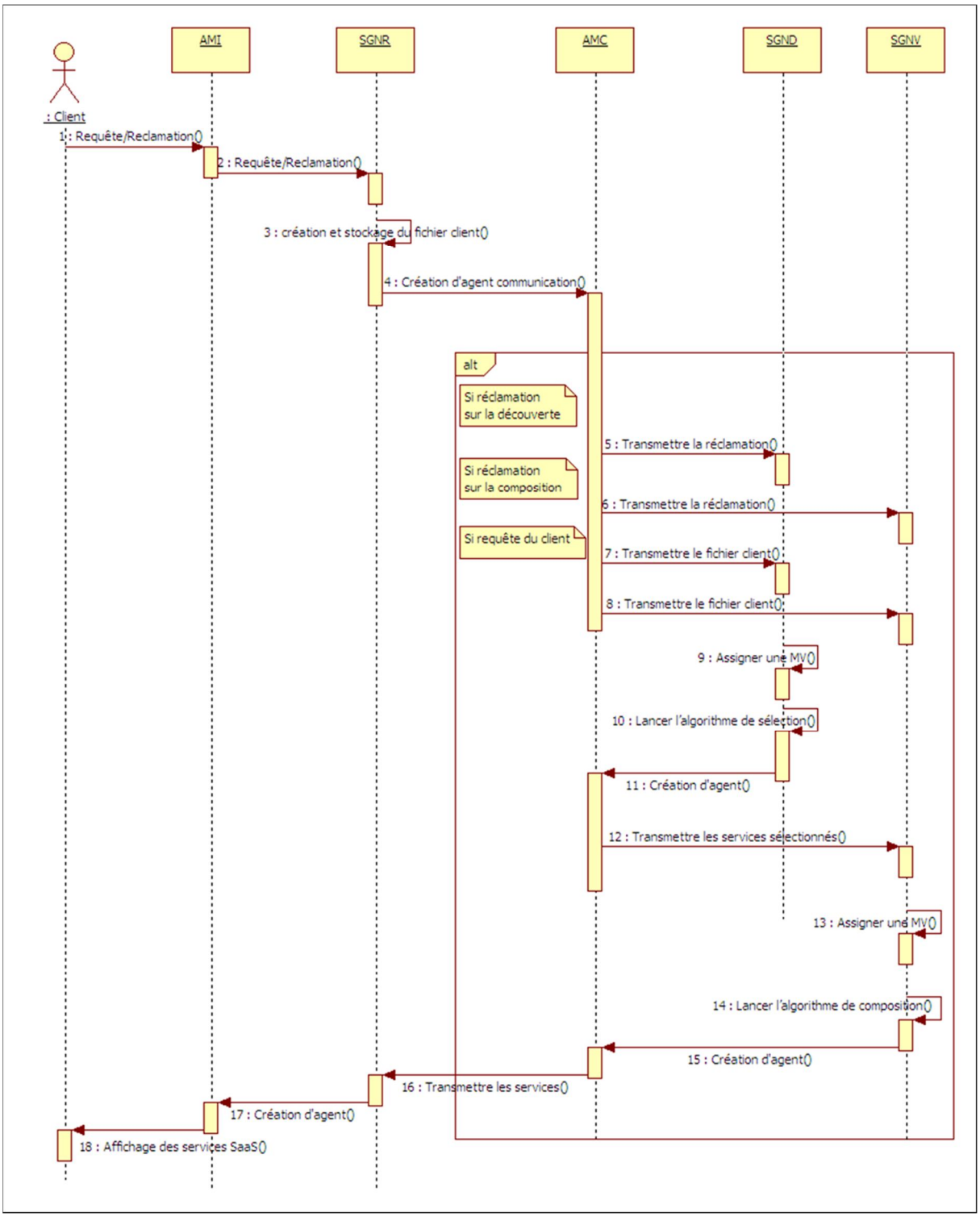

Figure V.7: Description UML des interactions générales entre composants du système.

V.5.1.2. Fonctionnement du nœud de réception

Le scénario du fonctionnement du nœud de réception est décrit par les étapes suivantes:

1. L'agent Interface apporte la requête ou la réclamation du client à l'agent administrateur.
2. Si c'est une requête, l'agent administrateur demande à l'agent d'infrastructure, d'assigner une machine virtuelle pour le traitement de la requête du client (sinon, l'agent administrateur, demande à l'agent d'infrastructure d'assigner une machine virtuelle pour l'opération d'indexation de la réclamation, ensuite c'est à l'agent statistique de lancer l'opération d'indexation).
3. L'agent administrateur, crée le fichier client avec son identifiant en utilisant le composant Software-as-a-Service, et le stock en utilisant le composant Storage-as-a-Service (si c'est une réclamation, l'agent administrateur crée un agent de communication afin d'envoyer la réclamation au nœud approprié ... si c'est une réclamation administrative, le composant Humman-Support-as-a-Services prend la réclamation pour la traiter).
4. L'agent administrateur crée deux agents de communication, pour envoyer le fichier client aux nœuds de découverte et de vérification.
5. Après le traitement de la requête par le nœud de vérification et classification, l'agent de communication apporte les résultats à l'agent administrateur du nœud de réception.
6. L'agent administrateur, identifie à l'aide du composant Software-as-a-Service, la requête par rapport au services composite, et cela en utilisant l'identifiant du client envoyé par le nœud de vérification et classification.
7. L'agent administrateur, crée un agent Interface pour afficher les services virtuels composites au client.

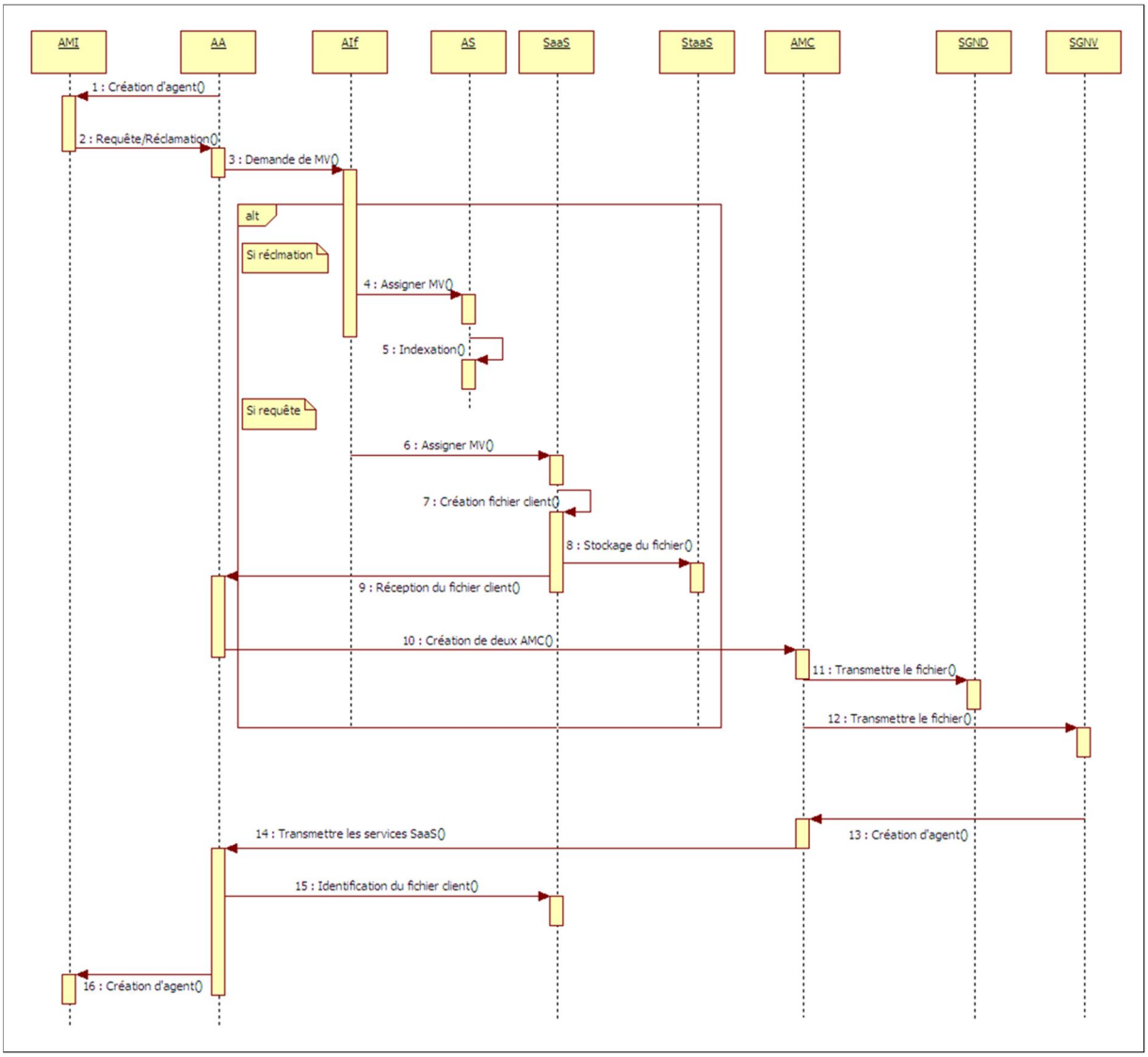

Figure V.8: Description UML des interactions entre les composants du nœud de réception.

V.5.1.3. Fonctionnement du nœud de découverte

Le scénario du fonctionnement du nœud de découverte est décrit par les étapes suivantes:

1. L'agent de communication apporte la requête ou la réclamation du client à l'agent administrateur.
2. Si c'est une requête, l'agent administrateur demande à l'agent d'infrastructure, d'assigner une machine virtuelle pour le traitement de la requête du client (sinon, l'agent administrateur, envoie la réclamation au composant Developer-as-a-Service, pour répondre au client).
3. L'agent administrateur, crée une zone de stockage pour les fichiers client avec leurs services sélectionnés, et cela en utilisant le composant Storage-as -a-Service.
4. L'agent de Prétraitement, lance le prétraitement de la requête, en utilisant le composant Discovery-as-a-Service.

5. L'agent de matching lance le mécanisme de découverte et sélection de service, sur la base de la requête du client et les arbres WSDL crée à chaque fois qu'un nouveau fichier WSDL est ramené sur le nœud de découverte, et cela en utilisant le composant Discovery-as-a –Service.
6. A Chaque fois qu'un nouveau service est sélectionné, l'agent administrateur le stocke dans la zone de stockage appropriée (avec son fichier client).
7. L'agent administrateur crée un agent mobile de communication, pour transmettre les services sélectionnés (associés à l'identifiant de la requête) au nœud de vérification et classification.
8. L'agent administrateur supprime le fichier client et les services sélectionnés du nœud de découverte, pour libérer l'espace de stockage.

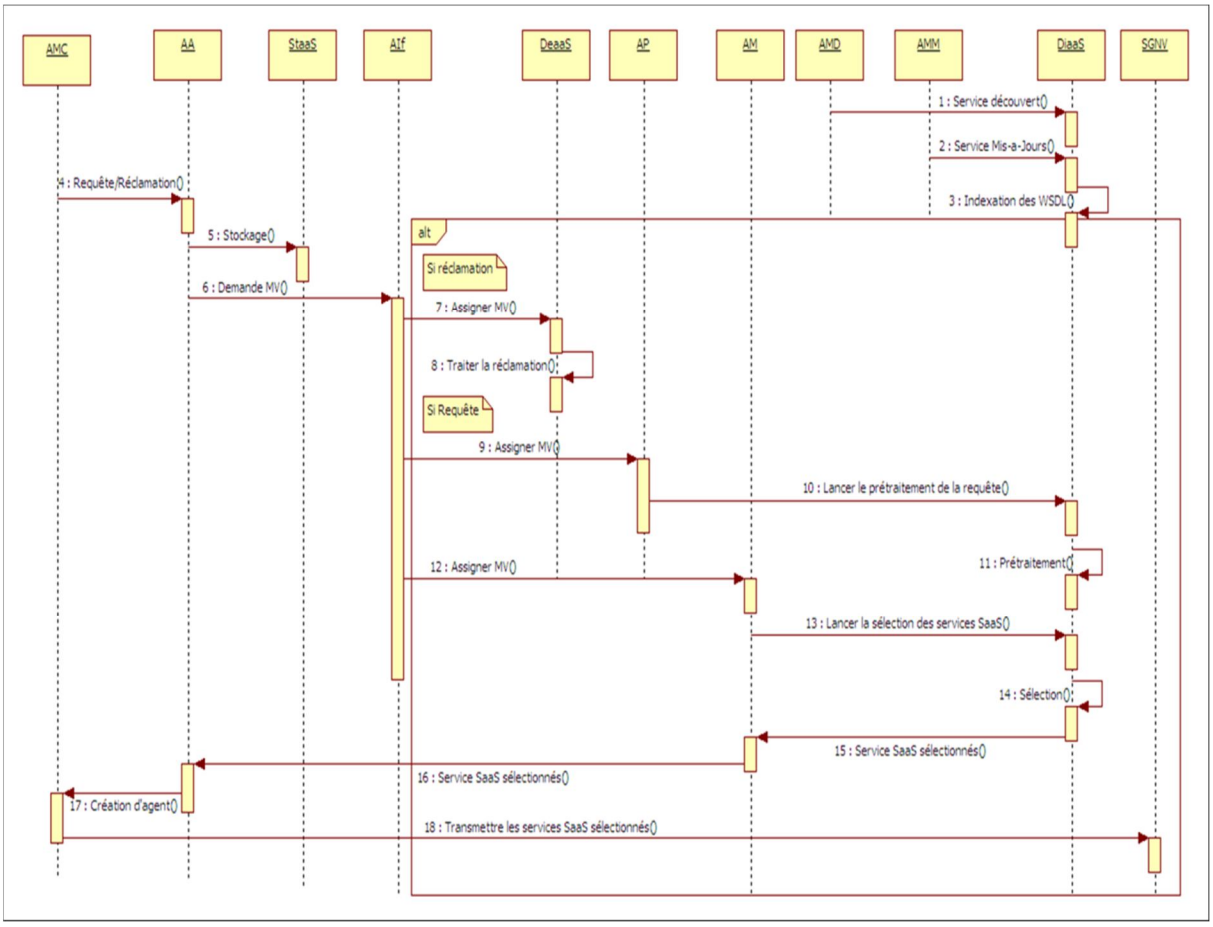

Figure V.9: Description UML des interactions entre les composants du nœud de découverte.

V.5.1.4. Fonctionnement du nœud de vérification et classification

Le scénario du fonctionnement du nœud de vérification et classification est décrit par les étapes suivantes:

1. L'agent de communication apporte le fichier du client et les services sélectionnés (associés à leur identifiant) à l'agent administrateur (si c'est une réclamation, l'agent

administrateur, envoie la réclamation au composant Developer-as-a-Service, pour répondre au client).

2. L'agent administrateur stocke le fichier client avec ses services sélectionnés, dans la même zone de stockage, en utilisant le composant Storage-as-a-Service.
3. L'agent administrateur, demande à l'agent d'infrastructure d'assigner une machine virtuelle afin de lancer la création des services virtuels composites.
4. L'agent de catégorisation utilise le composant Composition-as-a -Service pour décomposer la requête du client en sous-requêtes et pour mettre chaque service sélectionné dans sa catégorie ... en parallèle, l'agent de qualité classe les services Web sélectionnés en fonction de leurs paramètres QoS.
5. L'agent de composition, utilise le composant Composition-as-a-Service pour tester la connectivité des services atomiques sélectionnés, et cela en suivant les règles de connectivité décrit sur le chapitre précèdent.
6. L'agent administrateur, crée un agent de communication pour transmettre les résultats au nœud de réception.

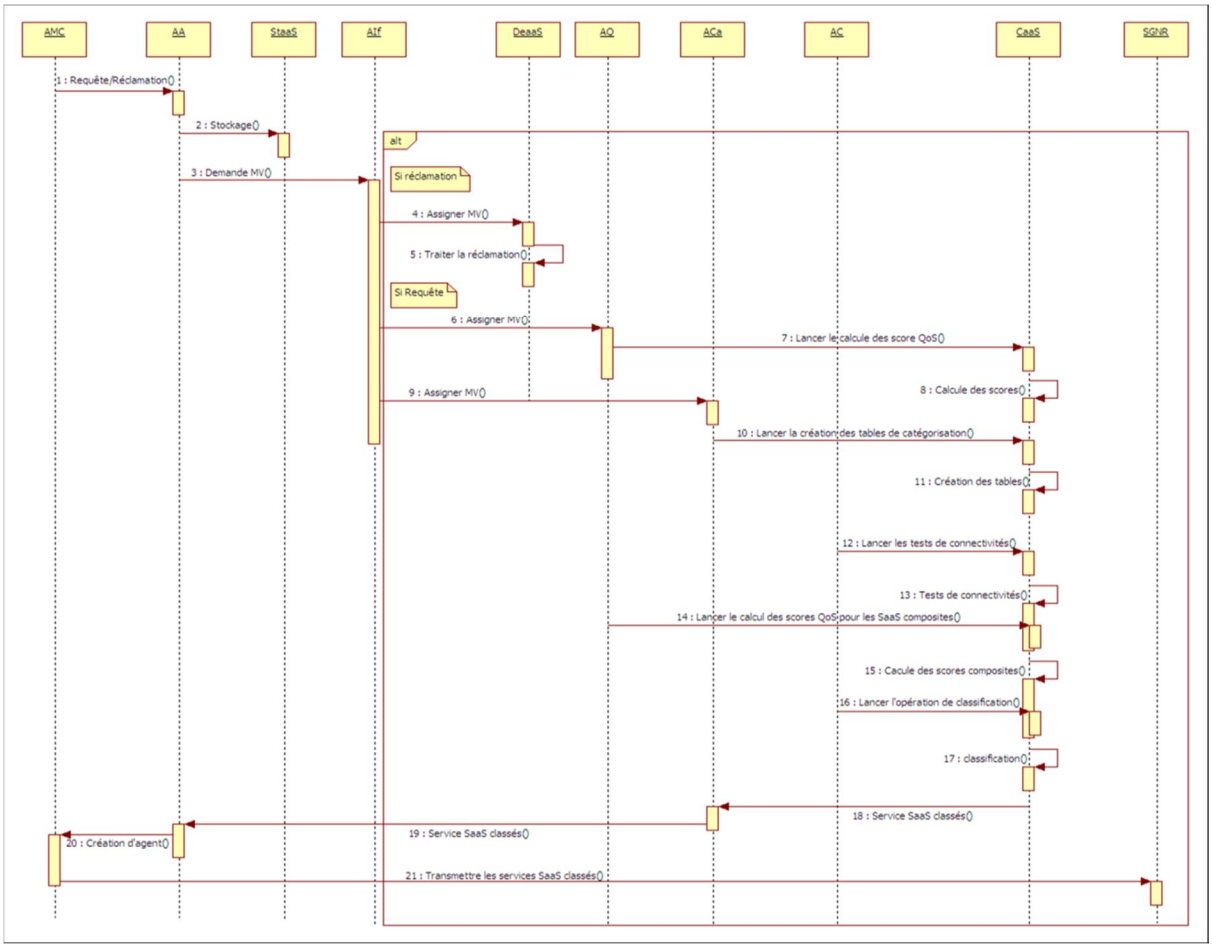

Figure V.10: Description UML des interactions entre les composants du nœud de vérification.

V.5.2. Interactions entre agents

Le comportement d'un Agent est défini comme un ensemble de tâches exécutées par un agent pour accomplir un but prédéterminé. Le comportement d'un agent est activé en réponse à un événement relative à son objectif. Un comportement peut être ajouté, supprimer ou arrêter, en fonction des besoins du système. Dans cette section, nous présentons les comportements et les interactions des agents du moteur proposé (Tableau V.1):

Agents	**Identificateur de comportement**	**Fonctions principale**
AA	*InitiateurNœud*	Pour soumettre les fichiers des clients au composant de chaque nœud.
	StockFichier	Pour enregistrer le fichier et les services sur chaque Nœud.
	DistributeurTâches	Pour gérer le composant SaaS du nœud de réception
	ContratRessources	Pour déployer et composé des ressources Cloud externe dont le nœud a besoin.
	MisàJourNœud	Pour mettre à jour l'ontologie WordNet, et l'index WSDL.
	GestionARD	Pour superviser l'opération de recherche de nouveaux services IaaS et SaaS
AIf	*GestionMV*	Pour gérer la virtualisation des ressources physiques sur chaque nœud.
	DistributeurMV	Pour distribuer les taches sur les machines virtuelles.
	PrédicteurBesoin	Pour prédire et composé les ressources internes sur chaque nœud.
AS	*IndexeurRQ*	Pour superviser l'opération d'indexation des réclamations et des questions des clients.
AP	*PréparateurDécouverte*	Pour Superviser l'opération de Prétraitement de la requête et des fichiers WSDL.
	DiaaS-AMInitiateur	Pour soumettre les mots clés de la requête et des fichiers WSDL à l'AM.
AM	*DiaaSInitiateur*	Pour soumettre les services atomiques sélectionnés au composant DiaaS.
	SélectionneurSS	Pour superviser l'opération de sélection des meilleurs services atomiques.

AQ	*CaaS*-ACa*Initiateur*	Pour soumettre les scores QoS des services atomiques à l'ACa.
	CalculateurQoS	Pour superviser l'opération de calcul des scores QoS atomiques et agrégatifs.
ACa	*CaaS-ACInitiateur*	Pour soumettre les tableaux de catégorisation a l'AC.
	CatégoriserSS	Pour superviser l'opération de création des tables de catégorisation.
AC	*CaaSInitiateur*	Pour soumettre les services sélectionnés au composant CaaS.
	VérificateurSS	Pour superviser l'opération de création des services virtuels composites.
	ClasseurSS	Pour superviser l'opération de classification des services atomiques et composites.
AMI	*InteractionClient*	Pour relier les clients avec le nœud de réception.
AMC	*LiaisonNœud*	Pour relier les nœuds du moteur SDCE-ACS ainsi que les ressources IaaS.
AMD	*ChercheurSS*	Pour chercher de nouveaux SaaS sur Internet.
	DistributeurAD	Pour coordonner l'opération de découverte des SaaS.
AMM	*MisàJourSS*	Pour soumettre les fichiers WSDL mis à jours, à l'AA du nœud de découverte.
AMR	*ChercheurRC*	Pour chercher des nouveaux IaaS sur Internet.
	DistributeurAR	Pour coordonner l'opération de découverte d'IaaS.
	ComposantsDéployer	Pour déployer les composants du moteur proposé sur les nœuds IaaS.

Tableau 0V.1: Résumé du comportement d'agents

Les agents de gestion des nœuds Cloud ainsi que les agents mobiles interagissent entre eux pour gérer : les ressources virtuelles de chaque nœud, et le mécanisme de découverte et composition de services. En plus, les AAs sont les seuls points d'accès à n'importe quel nœud du moteur proposé.

L'AA interagit avec les agent AP , AQ , l'AIf et AS et tous les agents mobiles et cela en adoptant : (i) *InitiateurNœud* : pour recevoir les requêtes et les réclamations des clients de l'IA , ensuite il soumet les fichiers des clients , les services sélectionnés, les services virtuels composites ou les réclamations aux agent : PA , QA (CaA et CA), et SA , des nœuds, respectivement , découverte , vérifications et réception, à l'aide de l'AMC , (ii) *StockFichier*:

pour superviser les opérations de stockage , (iii) *DistributeurTâches* : pour soumettre les tâches de créations des fichiers des clients , au composant SaaS du nœud de réception et SA , (iv) *ContratRessources* : pour composer les ressources Cloud nécessaires (ressources externes) dans le cas où l'AIf le demande, (v) *MisàJourNœud* : Pour recevoir les fichiers WSDL mise-à-jour par l'UA , et également mettre à jours l'ontologie WordNet quand une nouvelle version apparaît (sur le nœud de découverte) , (vi) *RDA Manager* : pour superviser la recherche de nouveaux services SaaS et de nouvelles ressources Cloud en utilisant les agents DA et RA.

L'AIf interagit avec les agents : AA, AP, AM, AQ, ACa et AC en adoptant : (i) *GestionMV*: pour créer les MV nécessaires pour le traitement des requêtes des clients, et composer des ressources Cloud(ressources interne), parmi celles disponibles, et cela quand les agent AA , AP , AM , AQ , ACa et AC ont en besoin pour exécuter leurs tâches , (ii) *DistributeurMV* : pour distribuer les tâches des agents AA , AP , AM , AQ , ACa et AC sur les machines virtuelles disponibles, (iii) *PrédicteurBesoin*: ils le font sur la base du taux d'utilisation des MVs et de l'espace de stockage utilisé par les agents : AA , PA , AM , AQ , ACa et AC .L'AIf composent les ressources Cloud nécessaires (ressources internes) pour des besoins futurs.

L'AS interagit avec les agents : AA ,AIf et AMC , en adoptant : (i) *IndexeurRQ*: pour recevoir les réclamations et les questions des clients à partir de l'AA , pour lancer l'opération d'indexation en utilisant le composant SaaS, la création des AMC pour transmettre les réclamations aux nœuds de découverte et de vérification, et pour demander des MVs supplémentaires de l'AIf en cas de besoin.

L'AP interagit avec les agents : AA, AIf et AM, en adoptant : (i) *PréparateurDécouverte* : Pour recevoir les requêtes des clients et les fichiers WSDL de l'AA, lancer l'opération de prétraitements de la requête, la création des arbres WSDL en utilisant le composant DiaaS, et demander des MVs supplémentaires de l'AIf en cas de besoin, (ii) *DiaaS-AMInitiateur*: pour soumettre les mots-clés de la requête et arbres WSDL aux AM.

L'AM interagit avec les agents : AP et AIf, en adoptant : (i) *DiaaSInitiateur*: Pour recevoir les mots-clés de la requête du client et des Arbres WSDL de l'AP, et les soumettre au composante DiaaS, (ii) *SélectionneurSS* : Pour lancer l'opération de sélection des services, et demander des MVs supplémentaires de l'AIf en cas de besoin.

L'AQ interagit avec les agents : AA, AIf et ACa, en adoptant: (i) *CaaS-ACaInitiateur*: Pour recevoir les poids des paramètres QoS introduit par le client, pré-classer les services atomiques basés sur leurs scores QoS, et les soumettre au composanteCaaS ,(ii) *CalculateurQoS* : Pour lancer l'opération de calcul des scores QoS, des services atomiques et composite en utilisant le composant CaaS, et pour demander des MVs supplémentaires de l'AIf en cas de besoin.

L'ACa interagit avec les agents : AA, AIf, et AQ, en adoptant: (i) *CaaS-ACInitiateur*: Pour recevoir les scores QoS de l'QA, les mots-clés de la requêtes et les services sélectionnés de l'AA, afin de les soumettre au composant CaaS , (ii) *CatégoriserSS*: pour lancer la création de tables de catégorisation (en utilisant le composant CaaS), et demander des MV supplémentaires de l'AIf en cas de besoin.

L'AC interagit avec les agents : AA et AIf en adoptant: (i) *CaaS-initiateur*: Pour recevoir les tables de catégorisation de l'ACa, les mots-clés de la requêtes et les services sélectionnés de l'AA, afin de les soumettre au composant CaaS , (ii) *VérificateurSS*: Pour lancer les tests de connectivité entre les services sélectionnés afin de choisir les meilleures chaînes de composition de services en utilisant le composant CaaS et demander des MVs supplémentaire de l'AIf en cas de besoin , (ii) *ClasseurSS*: pour classer les services atomiques et composites en utilisant le composant CaaS et demander des MVs supplémentaires de l'AIf en cas de besoin.

L'AI interagit avec l'AA en adoptant : (i) *InteractionClient* : Pour apporter les requêtes ou les réclamations des clients à l'AA, et afficher les services virtuels composites aux clients.

L'AMC interagit avec l'AA en adoptant: (i) *LiaisonNœud* : pour transmettre les fichiers et les réclamations des clients, les caractéristiques des ressourcesCloud, les services sélectionnés et les services virtuels composites entre les principaux nœuds du moteur proposé.

L'AMD interagit avec l'AA en adoptant : (i) *ChercheurSS*: pour la recherche de nouveaux services SaaS sur Internet, (ii) *DistributeurAD*: pour la Coopération avec les autres AMD afin d'éviter de chercher dans les mêmes régions d'Internet.

L'AMM interagit avec l'AA en adoptant : (i) *Mise-à-JourSS* : Pour mettre à jour les fichiers WSDL indexées dans le nœud de découverte.

L'AMR interagit avec l'AA et l'AIf en adoptant: (i) *ChercheurRC* : pour la recherche de nouveaux fournisseurs de services Cloud IaaS sur Internet, (ii) *DistributeurAD*: Pour la coopération avec les autres AMR afin d'éviter de chercher dans les mêmes Régions Internet, (iii) *ComposantsDéployer* : Pour déployer l'AIf dans le nœud IaaS cible, afin de tester son intégrité et sa fiabilité, et pour commencer le déploiement des composants du moteur proposé sur ce nœud.

Enfin, nous notons que les comportements des agents : AMR et AMD, sont très similaires, mais l'AMD cherche des nouveaux fournisseurs de services SaaS pour le compte des clients, alors que l'AMR cherche de nouveaux fournisseurs de services IaaS pour le compte du moteur proposé.

La figure V.11 illustre les interactions entre agents :

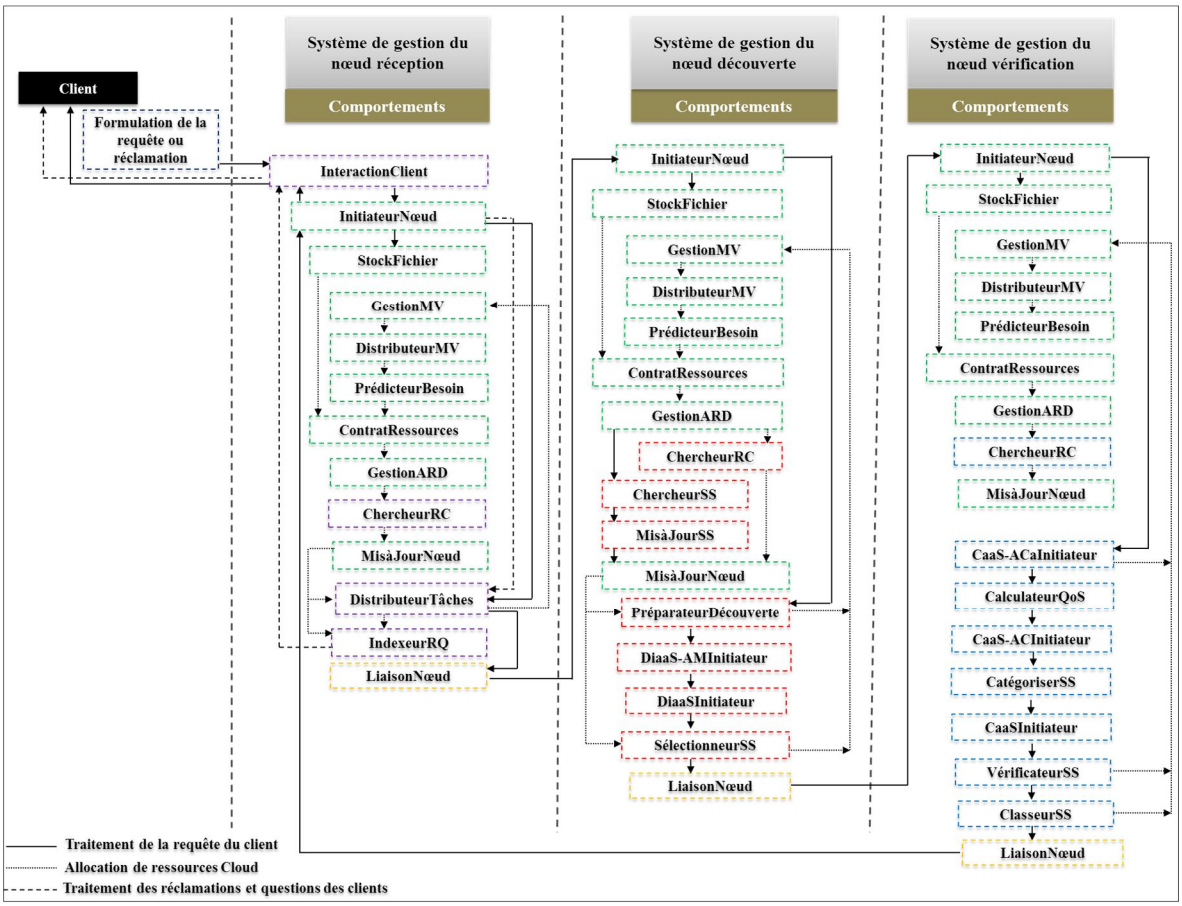

Figure V.11: Diagramme d'interactions entre agents.

V.6. Comportements des agents

Dans cette section, nous présentons les principaux comportements des agents régissant le fonctionnement des nœuds Cloud:

V.6.1. Comportements de l'Agent Administrateur

Les agents administrateurs de chaque nœud sont dotés d'un ensemble de comportements: le comportement *InitiateurNœud* est pour la réception des requête et des réclamations des clients, les services sélectionnés (cas de nœud de vérification) et les services virtuels composites (cas du nœud de réception), ensuite l'AA attend jusqu'à ce que les composants de chaque nœud traite les requêtes ou réclamations des clients pour créer un agent de communication qui transmet les information des clients (par information des clients, nous entendons : les fichiers ou réclamations des clients, les services sélectionné ou les services composites) au nœud suivants. Si le temps de latence (temps de traitement des requêtes) dépasse un certain seuil, l'AA vérifie l'agent responsable de la surveillance de cette tâche, si cet agent fonctionne normalement, l'AA vérifie l'AIF, si un agent est endommagé, l'AA le remplace par un autre agent, avec le même état de celui remplacé. Le comportement *StockFichier* est pour sauvegarder les fichiers des clients, les services sélectionnés et services virtuels composites, de manière temporaire jusqu'à ce que le client cible quitte le système. Si l'AA ne trouve pas un espace de stockage suffisant, il utilise l'espace de stockage du nœud de

sauvegarde, ensuite il consulte la TAAR pour trouver le nœud le plus proche qui peut fournir assez d'espace de stockage avec un coût minimal, par la suite il crée un agent mobile de ressources pour transmettre les informations des clients. Le comportement *DistributeurTâches* ne concerne que le nœud de réception, lorsque le temps de latence de la création des fichiers clients et des identificateurs dépasse un certain seuil, l'AA demande à l'AIf de lui offrir une nouvelle machine virtuelle pour créer une autre instance du composant SaaS, ensuite il lance la création des fichiers et identificateurs dans cette nouvelle instance. Si l'AIf ne peut pas créer une nouvelle MV, l'AA utilise le nœud de sauvegarde pour déployer la nouvelle instance SaaS, et il consulte la TAAR pour trouver le nœud le plus proche qui fournit des services de virtualisation à un coût minimal, ensuite il crée un agent mobile de ressources pour transmettre les codes nécessaires à déployer une nouvelle instance SaaS. Le comportement *ContratRessources* est pour composer des ressources (nécessaires pour les besoin de chaque nœud) à partir des nœuds IaaS externes, afin de déployer les services de chaque nœud; nous distinguons deux types de composition : composition horizontale, qui consiste à combiner des services hétérogènes (par exemple, ressources de stockage avec des CPU) et la composition verticale, qui consiste à combiner des ressources homogènes (par exemple, ajouté plus d'unités CPU). Lorsque l'AIf prévient l'AA que les ressources d'un nœud peuvent ne pas suffire à l'avenir, l'AA consulte la TAAR afin de chercher des nouveaux services de: virtualisation, stockage et calcul, pour créer un nouveau nœud Cloud. Il crée ensuite un agent mobile de ressources afin de déployer l'AIf sur ce nœud, pour tester l'intégrité et la fiabilité des ressources Cloud. Si les ressources fonctionnent bien l'AMR déploie les composants Cloud du nœud ciblé, sinon l'AMR retourne un rapport négatif à l'AA. Lorsque l'AIf effectue une demande à l'AA pour chercher des nouvelles ressources externes, il assigne à chaque demande un degré d'urgence , nous distinguons trois cas : (i) des besoins à long terme : cela signifie que le nœud cible peut avoir besoin de ressources supplémentaires au bout d'une heure ou plus , (ii) des besoin à durée moyenne : cela signifie que le nœud cible pourrait avoir besoin de ressources supplémentaires au bout de 30 minutes ou plus , (iii) des besoins à court terme: cela signifie que le nœud cible peut avoir besoin de ressources supplémentaires dans moins de 30 minutes. Outre, l'AA déploie automatiquement les composants des principaux nœuds dans les zones du monde qui sont dans une période de pointe, et cela pour réduire le temps de latence pour les clients. Le comportement *Mise-à-Jour-Nœud* est pour mettre à jour l'index WSDL du nœud de découverte (quand l'AMD apporte un nouveau fichier WSDL), la mise à jour de l'ontologie WordNet (quand une nouvelle version apparaît) et la Mise à jour de la TAAR (quand l'AMR apporte de nouvelles informations sur les fournisseurs de services IaaS). Le comportement *GestionARD* est pour la création et la distribution des agents AMR et AMD sur Internet, et la recherche de nouveaux fournisseurs de services IaaS et SaaS. Les agents AMR et AMD communiquent toujours leurs positions à l'AA, ce qui lui permet de les répartir sur les différentes régions d'Internet afin de couvrir un nombre maximal de fournisseurs de services.

Comportement 1 Initiateur Nœud

Input: (i) Réclamation ou questions des clients, ou (ii) requête du client ou (iii) Services découverts (iv) ou Services composites.
Output: (i) Identificateur du client, (ii) ou Réponse au client, (iii) ou Services découverts (iv) ou Services composites.

```
1: try
2:    switch (Nœuds principaux)
3:       case (Nœud de Réception):
```

```
4:              if (Réclamation ou questions des clients) then
5:                envois de messages à l'AS (Réclamation ou questions des clients);
6:              if (Temps de latence > seuil prédéterminé) then
7:                vérifier le statut de l'AS;
8:                if (l'SA est endommagé) then
9:                  créer un nouveau (AS) avec le même statut de l'AS original;
10:               else
11:                 vérifier le statut de l'AIf;
12:                 if (l'AIf est endommagé) then
13:                   créer un nouveau (AIf) avec le même statut de l'AIf original;
14:                   throw exception;
15:               recevoir la réponse du client;
16:             if (Requête du client) then
17:               envoi de la requête au composant SaaS;
18:               créer l'identificateur et le fichier du client;
19:               throw exception;
20:           break;
21:     case (Nœud de Découverte):
21:           recevoir le fichier du client;
22:           envois de messages  l'AP (requête du client, fichiers WSDL);
23:           if (temps de latence > seuil prédéterminé) then
24:             vérifier les statuts des agents AP et AM;
25:             exécution des instructions de 8 à 13 pour les agents AP et/ou AM;
26:           recevoir les services découverts;
27:           break;
28:    case (Nœud de Vérification et Classification):
29:          recevoir le fichier du client;
30:          recevoir les services découverts;
31:          envois de messages  aux agents AQ, ACa, et AC (requête du client, arbres
             WSDL);
32:          if (temps de latence>seuil prédéterminer) then
33:            vérifier les statuts des agents AQ, ACa, et AC;
34:            exécution des instructions de 8 à 13 pour les agents AQ, ACa, et/ou AC;
35:          recevoir les services composites;
36:          break;
37: catch (exception)
```

Algorithme V.1 : Initiateur global de nœud Cloud

Comportement 2 StockFichier

Input: (i) Réclamation ou questions des clients (ii) ou Requête du client (iii) ou Services découverts (iv) ou Services composites.
Output: (i) Espace de stockage.

```
1: rechercher un espace de stockage suffisant;
2: if,(trouver un espace de stockage suffisant) then
       // Cas de réclamations ou questions des clients
3:   stocker la réclamation et sa réponse dans la même zone de stockage jusqu'à ce que le
       composant Hsaad les supprime;
       // Cas de nœud de découvert
```

4: stocker les services découverts avec leur fichier du client dans la même zone de stockage jusqu'à ce que le nœud de vérification les reçoive;
// Cas du nœud de vérification et classification
5: stocker les services Composite avec leur fichier du client dans la même zone de stockage jusqu'à ce que le nœud de réception les reçoive;
// Cas de nœud de réception
6: stocker les services Composite avec leur fichier du client dans la même zone de stockage jusqu'à ce que le client quitte le moteur de découverte et composition;
7: **else**
8: envois de message à l'AIf pour créer un vue logique unifiée des espaces de stockage dispersé dans les disques durs des serveurs d'hébergement;
9: **if** (AIf ne trouve pas assez d'espace de stockage) **then**
10: stocker les informations du client dans le nœud de sauvegarde;
11: Consulté la TAAR pour trouver le plus proche nœud IaaS de stockage;
12: Créer un AMR pour transmettre les informations du client au nœud IaaS de stockage;

Algorithme V.2 : Mécanisme de gestion d'espace de stockage

Comportement 3 ContratRessources

Input: (i) Type de ressources dont le nœud a besoin (Espace de stockage, CPU, Virtualisation … etc.) (ii) capacités des ressources (nombre de CPU, volume d'espace de stockage … etc.) (iii) Degré d'urgence.
Output: (i) Ressources IaaS Composite
1: recevoir les besoins en ressources exprimés par l'AIf;
2: **try**
3: **Switch (**Degré d'urgence**)**
4: **case** (besoins à long terme):
3: vérifier les besoins des autres ressources;
4: **if** (il y a des besoins a durée moyenne ou à court terme) **then**
5: donner la priorité aux autres demandes de ressources Cloud;
6: **else**
7: **if** (composition horizontale) **then**
8: consulter La TAAR;
9: cherché des nœuds IaaS hétérogènes avec un minimum de cout;
10: **if** (l'AA ne trouve pas de noud IaaS hétérogène) **then**
11: Composer des ressources hétérogènes à partir des nœuds IaaS qui sont proches les uns des autres;
12: créer un AMR;
13: envoyer l'AMR au nœud IaaS ciblé;
14: **else**
15: consulter la TAAR;
16: cherché des nœuds IaaS homogènes avec un minimum de cout;
17: **if** (l'AA ne trouve pas de nœuds IaaS homogènes) **then**
18: composer les ressources Cloud hétérogènes à partir d'un seul nœud IaaS;
19: **if** (les capacités des ressources du nœud IaaS ne sont pas suffisantes) **then**
20: composer des ressources homogènes à partir des nœuds IaaS qui sont proches les uns des autres;
21: créer un AMR;
22: envoyer l'AMR au nœud IaaS ciblé;

23: **break;**
24: **case** (besoin à durée moyenne):
25: vérifier les besoins des autres ressources;
26: **if** (il y a des besoins à court terme) **then**
27: donner la priorité aux demandes à court terme;
28: **else**
29: exécution des instructions de 7 à 22 pour les besoins à durée moyenne;
30: mais l'AA, cherche les nœuds IaaS hétérogènes ou homogènes avec un cout et distance géographique minimal;
31: **break;**
32: **case** (besoin à court terme):
33: envois de messages à l'AIf du nœud de sauvegarde pour composer des ressources Cloud à partir de ce nœud;
34: exécution des instructions de 7 à 22 pour les besoins à court terme;
35: mais l'AA, cherche les nœuds IaaS hétérogènes ou homogènes avec la distance géographique minimale;
//la priorité est donnée à la distance géographique dans ce cas pour éviter les temps de transfert de données.
36: **break;**
37: **catch** (exception)
38: **if** (la zone ciblé se trouve dans une période de point) **then**
39: commencer les déploiements automatiques des composants des principaux nœuds, selon le temps que prend l'AMR pour transmettre les données et les codes nécessaires à la zone ciblée;

Algorithme V.3 : Compositeur de ressources IaaS

V.6.2. Comportements d'Agent d'infrastructure

Les agents d'infrastructures de chaque nœud sont dotés d'un ensemble de comportements: Le comportement *GestionMV* est pour la gestion des capacités des machines virtuelles en fonction de la taille des tâches à accomplir par chaque agent de gestion, par exemple la MV qui exécutent les tests de connectivité, a besoin de plus de ressources que la machine virtuelle qui exécute l'opération de classification. L'AIf détermine les capacités de la mémoire vive et le nombre de processeurs virtuels pour chaque MV. Si la RAM et CPU d'une machine virtuelle donnée sont entièrement occupées l'AIf crée une nouvelle MV pour transférer des données et des tâches de la première à la deuxième machine,et cela afin de réduire la charge du travail et optimiser le temps de réponse. Outre, l'AIf supervise le statut des ressources physiques et collecte les espaces de stockage libérés, afin de les présenter comme un espace logiquement unifié. Le comportement *DistributeurMV* assure la continuité des services de virtualisation, en remplaçant les machines virtuelles en échec, et en étendant les capacités des MVs lorsque les agents de gestion le demandent. Outre, l'AIf supervise la migration des données et les tâches d'une MV vers une autre et cela quand le temps de latence d'une machine virtuelle dépasse un certain seuil (pour réduire sa charge de travail). Le comportement *PrédicteurBesoin* est pour étudié les capacités des ressources physiques à soutenir la création de nouvelles machines virtuelles par rapport au taux d'utilisation des MVs, et à la croissance du nombre de tâches à exécuter. Si l'AIf estime que le nœud cible aura besoin de MV supplémentaires, il consulte la TAAR afin de trouver les ressources physiques disponibles sur ce nœud, si l'AIf ne trouve plus de ressources physiques, il envoie un message à l'AA afin de cherche de nouvelles ressources externes. Outre, l'AIf estime les besoins futurs

en espace de stockage du nœud cible, en fonction du taux d'utilisation de l'espace de stockage et l'espace disponible sur chaque disques de stockage.

Comportement 4 Gestion MV

Input: (i) demande d'obtention de nouvelle MV (ii) demande d'obtention de plus d'espace de stockage
Output: (i) Nouvelle MV (ii) vue unifiée logique d'espace de stockage
1: recevoir une demande de création de nouvelles MV des agents AA, AS, AP, AM, AQ, ACa, ou AC;
2: **if** (les ressources physiques sont disponibles) **then**
3: **if** (temps de latence de la MV< seuil prédéterminé) **then**
4: ajouter de nouvelles ressources virtuelles (CPU Virtuel par exemple);
5: throw exception;
6: **if** (la RAM ou le CPU virtuel sont entièrement occupé) **then**
7: créer un nouvelle MV;
8: **if** (Grande opération) **then** // une grande opération comme les tests de connectivité.
9: maximiser la taille de la RAM et le nombre de CPU;
10: **if** (Opération moyenne) **then** //une opération moyenne comme l'indexation des fichiers WSDL.
11: maximiser seulement le nombre de CPU;
// Puisque ce genre d'opérations a besoin d'exécuté un grand nombre d'instructions en parallèle.
12: **if** (Petite opération) **then** //une petite opération comme le prétraitement de la requête.
13: minimiser la taille de la RAM et le nombre de CPU;
14 : Transférer les données et codes à la MV nouvellement créer;
15: throw exception;
16: **else**
17: envois de message à l'AA pour obtenir plus de ressources;
18: throw exception;
19: à la fin de chaque période l'AIf vérifie les disques durs de chaque serveur pour collecter les espaces de stockages dispersés afin de les grouper sur une même partition logique;
20: recevoir une demande d'obtention de plus d'espace de stockage;
21: **if** (la partition unifiée logique est disponible et suffisante) **then**
22: fournir cette partition à l'AA;
23: **else**
24: redémarrer une nouvelle opération de collections d'espaces de stockage;
25: **if** (il y a assez d'espace de stockage disponible) **then**
26: exécuter les instructions de 19 à 22;
27: **else**
28: envoyer un message à l'AA, pour trouver des nœuds IaaS de stockage;
29: throw exception;

Algorithme V.4 : Mécanisme de gestion de ressources virtuelles

Comportement 5 Distributeur MV

Input: (i) MV en état d'échec (ii) MVentièrement occupé
Output: (i) Nouvelle MV (ii) MV avec des capacités étendues (ii) Migration de données.

1: à la fin de chaque période l'AIf vérifie les statuts des MVS et reçoit des demandes des agents : AA, AS, AP, AM, AQ, ACa, ou AC pour obtenir plus de ressources;
2: **if** (les ressources physiques sont disponibles) **then**
3: **if** (MV en état d'échec) **then**
4: créer une nouvelle MV;
5: déployer les composants Cloud dans la MV nouvellement créée;
6: transférer les donnée de la MV en état d'échec vers la MV nouvellement créée;
7: **if** (MV est entièrement occupé) **then**
8: **if** (temps de latence > seuil prédéterminé) **then**
9: exécuter les instructions: 4, 5, et 6;
10: **if** (temps de latence acceptable) **then**
11: augmenter la taille de la RAM selon la taille de l'opération à exécuter;
12: augmenter le nombre de CPU selon la taille de l'opération à exécuter;
// Grande, moyenne, ou petite opération.
13: **else**
14: envoyer un message à l'AA, afin d'obtenir plus de ressources externes;
15: throw exception;

Algorithme V.5 : Mécanisme d'affectation de taches sur les machines virtuelles

Comportement 6 Prédicteur Besoin

Input: (i) ressources disponibles des principaux nœuds
Output: (i) besoins en MVs (ii) besoins en CPU (iii) besoins en espace de stockage.
1: à la fin de chaque période, lAIf vérifie le temps de latence de chaque MV;
2: calculer le taux d'utilisation, avec la somme des temps de latence, divisée par le nombre de MVs (Gl_Lt);
3: calculer le nombre des CPU physiques utilisés (Nb_CPU);
4: calculer la taille d'espaces de stockage utilisé (ES);
5: **For** une période de tempsPré- déterminé **do**
6: **if** ((Gl_Lt- précédant(Gl_Lt)) est toujours > seuil prédéterminé) **then**
7: envoyer un message à l'AA pour chercher des ressources de virtualisation externe, avec un dégré d'urgence égal à « besoin à court terme »;
8: **if** ((Nb_CPU- précèdent (Nb_CPU)) est toujours > seuil prédéterminé) **then**
9: le nombre de CPU dont le système a besoin dans le futur, est la somme de (Nb_CPU- précèdent (Nb_CPU)) pour chaque vérification, soustrait du nombre de CPUs libres;
// Nous pouvons calculer le nombre de CPU en utilisant le pourcentage de croissance du nombre de CPUs utilisés, mais l'objectif est de maximiser les capacités de ressources physiques afin d'assurer une évolutivité massive pour les ressources du moteur SDCE-ACS.
10: **if** ((Gl_Lt- précèdent(Gl_Lt)) augmente et diminue de façon continuelle, en comparaison par rapport au même seuil prédéterminé) **then**
11: envoyer un message à l'AA pour chercher des ressources de virtualisation externes, avec un degré d'urgence égale à « besoin en durée moyenne »;
12: exécuter les instructions: 8 et 9;
13: **if** ((Gl_Lt- précédant(Gl_Lt)) est toujours = seuil prédéterminé) **then**
14: envoyer un message à l'AA pour chercher des ressources de virtualisation externes, avec un degré d'urgence égale à « besoin à long terme »;
15: exécuter les instructions: 8 et 9;

16: **if** ((ES - précèdent(ES)) est toujours > seuil prédéterminé) **then**
17: envoyer un message à l'AA pour chercher des ressources de stockage externe, avec un degré d'urgence égale à « besoin à court terme »;
18: l'espace de stockage dont le système a besoin au futur, est la somme des (SS - précèdent(SS)) pour chaque vérification, soustrait de la taille d'espace de stockage disponible;
// Nous pouvons, également, calculer l'espace de stockage nécessaire en utilisant le pourcentage de croissance d'espaces de stockage utilisés, mais l'objectif est de maximiser les capacités de ressources physiques afin d'assurer une évolutivité massive pour les ressources du moteur de découverte et composition.
19: **if** ((ES - précèdent(ES)) augmente et diminue de façon continuelle, en comparaison par rapport au même seuil prédéterminé) **then**
20: envoyer un message à l'AA pour chercher des ressources de stockage externes, avec un degré d'urgence égale à « besoin en durée moyenne »;
21: exécuter l'instruction 18;
22: **if** ((SS – précèdent (SS)) est toujours = seuil prédéterminé) **then**
23: envoyer un message à l'AA pour chercher des ressources de stockage externes, avec un degré d'urgence égale à « besoin à long terme»;
24: exécuter l'instruction 18;

Algorithme V.6 : Mécanisme de prédiction de besoins en ressources Cloud

V.6.3. Comportements d'Agent de Prétraitement

L'agent de prétraitement est doté d'un ensemble de comportements: Le comportement *PréparateurDécouverte* est pour la soumission des requêtes des clients, et des fichiers WSDL au composant DiaaS, qui est responsable du prétraitement des requêtes et de la création des arbresWSDL. Lorsque le temps de latence de l'opération de prétraitement dépasse un certain seuil, l'AP effectue une demande à l'AIf pour obtenir une nouvelle machine virtuelle afin de créer une autre instance du composant DiaaS, si l'AP obtient une nouvelle machine virtuelle, il lance le prétraitement de la requête dans la nouvelle instance DiaaS. Outre, l'AP choisit toujours l'instance DiaaS qui traite le moins de requêtes.

V.6.4. Comportements d'Agent de Matching

L'agent Matching est doté d'un ensemble de comportements: Le comportement *SélectionneurSS* est pour superviser la partie du composant DiaaS responsable du calcul du degré d'existence globale, et la sélection des services. Lorsque le temps de latence de l'opération de calcul du degré d'existence globale ou sélection des services dépasse un certain seuil, l'AM demande à l'AIF d'obtenir une nouvelle machine virtuelle afin de créer une autre instance du composant DiaaS. Si l'AM obtient une nouvelle machine virtuelle, il lance les opérations de calculs du degré d'existence globale, et de sélection des services dans la nouvelle instance DiaaS. En outre, l'AM choisit toujours l'instance DiaaS à laquelle il reste le moins d'arbres WSDL.

V.6.5. Comportements d'agent de Qualité

L'agent de la qualité est doté d'un ensemble de comportements: Le comportement *CalculateurQoS* est pour superviser la partie du composant CaaS, responsable du calcul des

scores QoS des services atomiques et composites. Lorsque le temps de latence des opérations de calculs des scores QoS dépasse un certain seuil, l'AQ effectue une demande à l'AIf pour obtenir une nouvelle machine virtuelle, afin de créer une autre instance du composant CaaS. Si l'QA obtient une nouvelle machine virtuelle, il lance les opérations de calculs des scores QoS dans la nouvelle instance CaaS. En outre, l'AQ choisit toujours l'instance CaaS à laquelle il reste le moins de paramètre QoS à traiter.

V.6.6. Comportements d'Agent de catégorisation

L'agent de catégorisation est doté d'un ensemble de comportements: Le comportement *CatégoriserSS* est pour superviser la partie du composant CaaS responsable de la création des tables de catégorisation. Lorsque le temps de latence de l'opération de création des tables de catégorisation dépasse un certain seuil, l'Aca effectue une demande à l'AIf pour obtenir une nouvelle machine virtuelle et cel afin de créer une autre instance du composant CaaS. Si l'ACa obtient une nouvelle machine virtuelle, il lance les opérations de création des tables de catégorisation dans la nouvelle instance CaaS. En outre, l'ACa choisit toujours l'instance CaaS qui a les plus grandes capacités virtuelles.

V.6.7. Comportements d'Agent de composition

L'agent de composition est doté d'un ensemble de comportements: Le comportement *VérificateurSS* est pour superviser la partie du composant CaaS, responsable d'effectuer les tests de connectivité. En plus, l'AC choisit toujours l'instance CaaS à laquelle il reste le moins de tests de connectivité. Le comportement *ClasseurSS*, est pour superviser la partie du composant CaaS responsable de la classification des services atomiques et composites. En outre, l'AC choisit toujours la MV à laquelle il reste le moins de services virtuels composites à classé. Lorsque le temps de latence dépasse un certain seuil, l'AC effectue une demande à l'AIf pour obtenir une nouvelle machine virtuelle et cela afin de créer une autre instance du composant CaaS. Si l'AC obtient une nouvelle machine virtuelle, il lance les tests de connectivité ou l'opération de classification sur la nouvelle instance CaaS.

Le comportement de l'AC ressemble beaucoup aux comportements des agents: AA, AP, MA, AQ et ACa en matière de gestion des composants : SaaS, DiaaS et CaaS, c'est pourquoi nous ne présentons que le comportement *ClasseurSS*:

Comportement 7 Classeur SS

Input: (i) services Atomiques et/ou composites
Output: (i) Nouvelle instance CaaS (ii) Services classés.
1: recevoir les services atomiques et/ou composites;
2: choisir l'instance CaaS à laquelle il reste le moins de service à classer;
3: **if** (au moins, le temps de latence d'une instance CaaS < = seuil prédéterminé) **then**
3: lancer l'opération de classification, et cela en initialisant le composant CaaS a l'aide des services atomiques et/ou composites;
4: classer les services en suivant le mécanisme décrit dans le chapitre 5;
5: **else**
6: envoyer un message à l'AIf pour obtenir une nouvelle MV;
7: quand l'AC obtient sa MV, il déploie une nouvelle instance du composant CaaS;

8: envoyer un message à l'AA, pour obtenir l'accès à une instance CaaS dans le nœud de sauvegarde;
9: exécuter les instructions: 3, 4, 5, 6, et 7;
// Les instructions 9 et 11 sont exécutées en parallèles
10: throw exception;

Algorithme V.7 : Gestion des ressources Cloud pour la classification de services SaaS

V.6.7. Comportements d'agent mobile de Mise à jour

L'agent mobile de Mise à jour est doté: du Comportement *MisàJourSS* qui sert à mettre à jour l'index WSDL du nœud de découverte. L'AMM encapsule, à la fin de chaque période de temps, un certain nombre de fichiers WSDL indexés dans le nœud de découverte, puis il migre vers l'emplacement des fournisseurs de services SaaS ciblés, pour parser leurs fichiers WSDL afin de détecter d'éventuelles différences entre les WSDL encapsulées et celles publiées. Si l'AMM détecte des différences, il se clone pour transmettre le nouveau fichier WSDL au nœud de découverte et l'AMM original continue de visiter les autres fournisseurs de services.

V.6.8. Comportements des agents mobiles de Découvertes et de Ressources

Les agents mobiles de Découverte et Ressources sont dotés d'un ensemble de comportements: Les comportements *ChercheurSS* et *ChercheurRC* sont pour la recherche de nouveaux fournisseurs services SaaS et IaaS.L'AA crée à chaque fois, un ensemble d'AMD et AMR, puis ils envoit chaque agent mobile sur sa région Internet. Quand un AMD ou AMR trouve un nouveau fournisseur de services SaaS ou IaaS il se clone pour transmettre le fichier WSDL ou les caractéristiques IaaS trouvées, et l'agent original continue de visiter les autres fournisseurs de services. Les comportements *DistributeurAD* et *DistributeurAR* sont pour communiquer les emplacements des agents AMD et AMR aux AA, ce qui permet d'optimiser la distribution de ces agents mobiles sur Internet. Outre, quand un AMD rencontre un autre AMD, l'AMD qui est dans sa région Internet détruit l'autre AMD, et cela pour éviter d'avoir des conflits et des redondances (les AMR ont le même comportement).De plus, l'agent mobile ressources est doté du comportement *ComposantsDéployer* pour déployer automatiquement les composants du système proposé sur les nœuds IaaS. Quand un nœud a besoin de ressources externes supplémentaires, l'AMR encapsule les codes nécessaires au déploiement de l'AIf et migre vers le nœud IaaS cible, afin de tester le fonctionnement des ressources Cloud. Nous distinguons deux cas, si l'AIf constate que les ressources fonctionnent bien, l'AMR utilise les outils de déploiements automatiques du nœud IaaS afin de déployer le système de gestion et les composants Cloud (SaaS, DiaaS et AAC), si ce n'est pas le cas, l'AMR transmet un rapport négatif à l'AA.

V.6.9. La complexité des comportements des agents

Afin de calculer la complexité des comportements d'agents, nous utilisons deux mesures:

V.6.9.1.Complexité temporelle

Tous les comportements des agents ont une complexité temporelle « linéaire » *O(n)* (Ceci peut être déterminé par une simple inspection des comportements présentés ci-dessus), a l'exception des :

- *Comportement3*:la complexité temporelle de ce comportement dépend du flux de travail associé à l'IaaS composite dont le nœud a besoin. Toutefois, dans le cas d'un IaaS atomique, la complexité temporelle est « linéaire » *O(n)*.
- Comportement6: la complexité temporelle de ce comportement est « quadratique » $O(n^2)$.
- Comportement7: la complexité du temps dépend du flux de travail associé aux services virtuels composites affichés aux clients. Toutefois, dans le cas de services atomiques la complexité temporelle est « linéaire » *O(n).*

V.6.9.2.Complexité des communications entre agents

Lacomplexité des communications entres agents est calculée par le nombre de messages échangés entre eux, dans le cas le plus pire:

- Agents administrateurs: Pour *q* requêtes des clients, les agents administrateurs de chaque principal nœud Cloud envoient cinq messages aux agents AS, AP, AM, ACa et AQ, afin de les initier avec les requêtes ou la réclamation des clients. Ensuite, l'AA envoie *SM* messages à l'agent d'infrastructure afin de récupérer plus d'espace de stockage. Si l'AIf ne trouve pas assez d'espace de stockage, l'AA envoie *IM* messages à l'AMR afin d'obtenir plus d'espaces de stockage à partir des nœuds IaaS. En conséquence, dans le pire des scénarios, l'agent administrateur de chaque nœud envoie:

  $$3q((SM + IM + 4) + (SM + IM + 2) + (SM + IM) + (SM + IM - 2) + (SM + IM - 4) + ... + 4) \quad \text{Messages.} \qquad (20)$$

  Lorsque *q*, *SM*, et *IM*, tendent vers l'infini, le nombre de messages échangés dans le pire scénario, est délimité par $6n^2+12n$ messages.

- Agents d'infrastructure: Pour traiter *q* requêtes des clients, les agents d'infrastructure de chaque principal nœud Cloud envoient *3q* messages à l'AA afin d'obtenir plus de d'espace de stockage, Unité de traitement ou et d'infrastructure composite. En conséquence, dans le pire des scenarios, les Agents d''infrastructure de chaque nœud envoient:

  $$q(3q + (3q-1) + (3q-2) + ... + (2) + (1)) = q(3q(3q+1)/2)) = 3q^2(3q+1)/2 \quad \text{Messages} \qquad (21)$$

  Lorsque *q* tend vers l'infini, le nombre de messages échangés dans le pire scenario est délimitée par $(9n^3+3n^2)/2$ messages.

- Les agents responsables de l'exécution du mécanisme de découverte et composition: Pour traiter *q* requêtes des clients, les agents chargés de l'exécution du mécanisme de découverte et composition (AP, AM, AQ, ACa, et AC) envoient *5q* messages à l'AIf et *5q* messages à l'AA. Outre, l'AP envoie *TM* messages à l'AM et l'AM envoie *SEM*

messages à l'AA (en fonction du nombre d'Arbres WSDL et des services sélectionnés). L'AQ envoie *AAM* messages à l'Aca et l'Aca envoie *ACaM* messages à l'AC, et l'AC envoie *ACM* messages à l'AA (en fonction du nombre de services atomiques et/ou composites). En conséquence, dans le pire des scenarios, les agents AP, AM, AQ, ACa, et AC envoient:

$$
\begin{aligned}
&10q + q((TM + SEM + AAM + ACaM + ACM) + \\
&(TM + SEM + AAM + ACaM + ACM - 5) + \\
&(TM + SEM + AAM + ACaM + ACM - 10) + \\
&(TM + SEM + AAM + ACaM + ACM - 15) + ... + \\
&(2) + (1)) \\
&= 10q + q((TM\ (TM + 1) + SEM\ (SEM + 1) + \\
&AAM\ (AAM + 1) + ACaM\ (ACaM + 1) + ACM\ (ACM + 1)) / 2) \\
&= q(((TM\ (TM + 1) + SEM\ (SEM + 1) + \\
&AAM\ (AAM + 1) + ACaM\ (ACaM + 1) + ACM\ (ACM + 1)) / 2) + 10) \text{ Messages}
\end{aligned}
\tag{22}
$$

Lorsque *q*, *TM*, *SEM*, *AAM*, *ACaM* et *ACM*, tendent vers l'infini, le nombre de messages échangés, dans le pire des scénarios, est délimité par $5((1/2)n^3+(1/2)n^2+2n)$ messages.

V.7. Conclusion

La contribution de ce chapitre, est que c'est l'une des premières architectures qui produit une approche Cloud basé agent pour la gestion des ressources virtuelles et logiques dans le cadre de la découverte et composition de services SaaS.

En outre, nous avons proposé un ensemble de comportement d'agents, qui représente les mécanismes de gestion de ressource Cloud, et qui permettent l'exécution des algorithmes de matching présentés dans le chapitre précèdent.

Ainsi le prochain chapitre présente le prototype qu'on a développé afin d'évaluer les performances (temps d'exécution et coût d'utilisation) des algorithmes présentés dans le chapitre 4 ainsi que les mécanismes présentés dans ce chapitre.

VI
Résultats Expérimentaux

VI.1. Introduction

Dans ce chapitre nous présentons le prototype développé à base des algorithmes de matching proposés dans le chapitre 4 ainsi que l'architecture proposée dans le chapitre 5.

Nous évaluons également l'approche proposée en utilisant quelques mesures qui nous permettent de situer le système proposé par rapport à quelques méthodes célèbres et récentes dans le domaine.

Ce chapitre est organisé comme suit :
Dans la section 2 nous présentons les outils utilisés pour l'implémentation du prototype. Dans la section 3 nous présentons les objectifs des expérimentations menées sur le prototype implémenté .Dans la section 4 nous présentons les mécanismes utilisés pour évaluer l'approche proposée. Dans la section 5, nous présentons le prototype. Dans la section 6, nous décrivons le modèle d'évaluation de performances adoptées, ainsi que les résultats obtenus pour chaque mesure d'évaluation. Enfin, la section 7 est la conclusion de ce travail.

VI.2. Outils et Plateformes Utilisés

Afin de faciliter le développement de notre application, nous avons utilisé l'environnement de développement intégré Eclipse Indigo basé sur le langage JAVA. Nous utilisons également la plate-forme JADE (version 4.1) pour le développement des agents, avec le langage de communication FIPA-ACL. Enfin, Nous utilisons, la plate-forme Cloudsim pour la simulation des ressources virtuelles ainsi que les taches exécutées par notre système.

VI.2.1. Simulateur Cloud

Un des grands problèmes de développement d'applications sur le Cloud est l'environnement dans lequel nous pouvons reproduire les tests. Faire des tests sur un environnement Cloud réel comme Amazon EC2, pourrait être très coûteux. Mais avec la simulation, les développeurs peuvent tester et optimiser leurs applications Cloud sans avoir à payer quoique ce soit, et même les clients peuvent tester les performances de leurs services de façon reproductible, et sur un environnement contrôlable et gratuit, avant de mettre les applications sur l'environnement Cloud réel.

Un simulateur Cloud est un outil qui permet de manipuler des Datacenter avec leurs serveurs de base et tout le hardware nécessaire à leur bon fonctionnement (CPUs, Disque dur, RAM … etc.), afin de simuler la création et distribution de taches sur des machines virtuelles.

Dans la validation que nous présentons sur cette thèse, nous utilisons le simulateur Cloudsim.

VI.2.1.1. Plate-forme Cloudsim

Cloudsim est un outil puissant de modélisation, simulation et expérimentation qui sert au développement d'infrastructure Cloud, il permet également aux développeurs de tester leurs services et applications Cloud, sans se préoccuper des détails de bas niveau liés aux infrastructures Cloud :

Les principales fonctionnalités de Cloudsim sont [149, 150]:

- Supporter la modélisation et la simulation des Datacenter Cloud à grandes échelle.
- Supporter la modélisation et la simulation des serveurs virtuels d'hébergement d'application, avec possibilité de personnaliser la politique de fourniture des ressources d'hébergements aux machines virtuelles.
- Supporter la modélisation et la simulation de ressources virtuelles et logiques pour le test et l'optimisation de la consommation d'énergie sur le Cloud.
- Supporter la modélisation et la simulation des différentes topologies réseaux, Datacenter et échange de messages entre applications Cloud.
- Supporter la modélisation et la simulation des systèmes Cloud fédérés.
- Supporter l'insertion dynamique de nouveaux éléments de simulation, avec des capacités d'arrêt et de reprise de simulation.
- Supporter tout type et politiques d'allocation de ressources pour la création de machines virtuelles, et tout type et politiques d'allocation de ressources pour l'hébergement d'application et de services Cloud.

VI.2.1.2. CloudAnalyst

Avec Cloudsim, il est très difficile de quantifier certains paramètres importants pour l'évaluation de l'efficacité des services Cloud, tels que la localisation géographique des Datacenter, qui permet de calculer la distance entre eux et leurs clients, l'impact du nombre d'utilisateurs simultanés, et l'implication de la topologie des réseaux sur la gestion des ressources.

CloudAnalys est un outil de simulation qui permet aux développeurs d'exécuter et de tester les différentes simulations de façon reproductible, en prenant en considération les différents paramètres suscités.

CloudAnalys est équipé d'une interface utilisateur graphique (GUI) qui facilite sont utilisation (voir figure 6.1), et qui permet aux développeurs de mettre en place des expériences de façon rapide et facile. Outre, CloudAnalyst permet de séparer entre l'aspect simulation et l'aspect programmation lors du développement des paramètres de chaque expérimentation. De cette façon le développeur peut se concentrer sur la complexité de la simulation au lieu de passer la majeure partie de son temps sur les aspects techniques de la programmation à l'aide d'un Toolkit de simulation [151].

Figure VI.1: Interface graphique du simulateur CloudAnalyst.

VI.2.2. Plate-forme JADE

Afin d'assurer un développement rapide est efficace des agents qui composent notre système, nous utilisons la plateforme de développement des systèmes multi-agent JADE. JADE regroupe un ensemble d'outils et d'API qui permettent la construction et la mise en service d'agents sur un contexte bien spécifique.

VI.2.2.1. Présentation générale

JADE est une plateforme implémentée complètement avec le langage JAVA. Ce Framework est destiné à faciliter le développement des systèmes multi-agents en conformité totale avec le standard FIPA. La plateforme JADE fournit une interopérabilité sans limite pour les applications quelles prend en charge, car cette plateforme est indépendante du système d'exploitation ainsi que du matériel sur laquelle elle est implémenté [152]. JADE est composée de trois principaux modules qui dépendent directement des normes FIPA :

- **DF :** pour Dirictor Facilitor en anglais, qui sert à la fourniture d'un service de « page jaune » à toute la plateforme JADE.
- **ACC :** pour Agent Communication Channel en anglais, qui représente l'outil de gestion de communication entre agents, pour la plateforme JADE.
- **AMS :** pour Agent Management System en anglais, qui représente l'outil de gestion (authentification, supervision, accès … etc.) des agents de la plateforme JADE.

VI.2.2.2. Architecture logicielle

Puisque JADE est une plateforme basée sur les standards FIPA, l'architecture de cette plateforme est également basée sur l'architecture proposée par FIPA. AMRM (pour Agent Management Reference Model en anglais) est le modèle de base de l'architecture de la plateforme JADE proposée par FIPA. Chaque module qui compose l'architecture de la plateforme JADE est présenté sous forme de service, ce qui permet aux agents de bénéficier d'une plateforme orientée service, afin de faciliter la communication et la collaboration entre eux.

Les principaux modules qui composent l'architecture JADE sont : DF, AMS et également le MTS (pour Massage Transport Service en anglais) qui sert de moyen pour la communication entre plusieurs plateformes JADE.

Afin d'assurer un fonctionnement efficace des agents sur la plateforme JADE, cette dernière utilise :

- **AID :** pour Agent Identifier en anglais, afin de distinguer et d'identifier chaque agent.
- **DF :** qui joue le rôle d'un annuaire servant à enregistrer les compétences de chaque agent. Les pages jaunes fournis par ce service, sont destinées à mettre en relation les différents agents fonctionnels sur la plateforme JADE, et cela pour qu'un agent puisse consulter et interroger ce service, afin d'obtenir des informations sur les compétences des autres agents et afin d'assurer une bonne collaboration entre eux.
- **AMS :** joue le rôle d'un annuaire pour l'enregistrement des adresses de transport des différents agents de la plateforme. Le but c'est de fournir un service de « pages blanches » afin de mettre en correspondance les agents avec l'AID, pour faciliter leur contrôle et supervision.

VI.2.2.3. Langage de communication

Le langage de communication FIPA-ACL (pour Agent Communication Langage) est le langage adopté par la plateforme JADE. De façon générale, la communication entre agents est en mode asynchrone, et elle est mise en œuvre en utilisant la classe ACLMessage. On distingue deux cas d'utilisation de la classe ACLMessage:

- Envoi de messages : quand un agent souhaite envoyer un message, il doit créer un Objet du type ACLMessage, ajouter les paramètres qu'il souhaite envoyés, et appeler la méthode send() pour envoyer son message.
- Réception de messages : quand un agent souhaite recevoir un message, il doit faire appe là la méthode recieve() ou blocking receive().

VI.2.2.4. Mobilité

Afin de permettre aux agents de migrer d'une plateforme à une autre ou d'un container à un autre, selon le type de mobilité visé par chaque agent mobile, la plateforme JADE utilise deux types de services:

- Mobilité Intra-plateforme : La plateforme JADE utilise un Service de gestion de la mobilité des agents (Agent Mobility Service en anglais) pour implémenter ce type de mobilité. Avec la Mobilité Intra-plateforme, l'agent mobile a la faculté d'émigrer d'un container à un autre, mais dans la même plateforme seulement, en utilisant la méthode doMove(). Outre, les méthodes beforeMove() et afterMove() servent comme moyen de déterminer les actions à entreprendre par l'agent mobile avant ou après la migration.

- Mobilité Inter-plateforme : La plateforme JADE utilise l'IPMS (pour Inter-Platform Mobility Service en anglais) pour implémenter ce type de mobilité. Avec la Mobilité Inter-plateforme, l'agent mobile a la faculté d'émigrer d'une plateforme à une autre, en utilisant la méthode move() et power-up (), La première méthode sert à l'émigration de l'agent avec ses données et son code, et la seconde méthode sert à activer l'agent une fois la migration effectuée.

VI.3. Objectifs et ajustements des Expérimentations

Dans cette section nous présentons les objectifs que nous désirons atteindre à partir des différentes expérimentations que nous avons effectuées à partir du prototype développé. Nous présentons également les paramètres que nous avons utilisés pour ajuster nos expérimentations.

VI.3.1. Objectifs

Dans ce chapitre nous évaluons les différents algorithmes proposés dans les chapitres 5 et 6 à travers la simulation du système proposé. Afin d'avoir une vision comparative globale du système proposé, nous comparons le temps de réponse et les couts d'utilisation du système proposé avec les méthodes linéaire et aléatoire, et cela en variant le nombre de clients qui y accèdent et utilisent simultanément le système proposé.

Les expérimentations visent à mettre en évidence la façon, grâce à laquelle, les améliorations apportées par notre système de gestion de ressources ainsi que notre algorithme de découverte et composition, réduisent le temps de réponse et les coûts d'utilisations.

VI.3.2. Ajustement et réglage de paramètres

Tous d'abord nous définissons les caractéristiques des Datacenter qui représentent les principaux nœuds du moteur:

1. Le premier Datacenter est composé de quatre serveurs d'hébergement d'applications et il représente le nœud de réception : (i) les premiers serveurs hébergent le composant Software-as-a-Service avec l'agent administrateur , (ii) le second serveur héberge le composant Storage-as-a-service afin de stocker les fichiers et les résultats retournés aux clients , (iii) le troisième serveur héberge les comptes des administrateurs humains, le mécanisme d'indexation (pour indexer les réclamation et les questions des clients) et l'agent statistique, (iv) le dernier serveur héberge le nœud de sauvegarde qui permet de stocker les données et les codes du nœud de réception.

2. Le deuxième Datacenter est composé de neuf serveurs d'hébergement d'applications, et il représente le nœud de découverte : (i) le premier serveur héberge l'agent administrateur et la partie de la composante Discovery-as-a-service qui reçoit les fichiers des clients , (ii) les deuxième et troisième serveurs hébergent l'agent de pré traitement , et la deuxième partie du composant Discovery-as-a-service responsable de la création des index WDSL , (iii) les quatrième et cinquième serveurs hébergent l'agent de matching , et la troisième partie du composante Discovery-as-a-service responsable de l'exécution du mécanisme de découverte et sélection de SaaS , (iv) les sixième et septième serveurs hébergent le composant Storage-as-a-service ,afin de stocker les fichiers des clients et les arbres WSDL qui correspondent aux requêtes , (v) le huitième serveur héberge les comptes des développeurs humains, et stocke les réclamations et les questions des clients , (vi)le dernier serveur héberge le nœud de sauvegarde qui permet de stocker les données et les codes du nœud de découverte .

3. Le troisième Datacenter est composé de sept serveurs, et il représente le nœud de vérification et classification : (i) le première serveur héberge l'agent administrateur et la première partie du composante Composition-as-a-Service , qui reçoit les fichiers des clients et les arbres WSDL sélectionnés, (ii) le second serveur héberge l'agent de

qualité et la deuxième partie du composant Composition-as-a-service responsable du calcul des scores QoS des service atomiques et composites , (iii) les troisième et quatrième serveurs hébergent l'agent de catégorisation, l'agent de composition, et la troisième partie du composant Composition-as-a-service responsable de la création et classification des services composites , (iv) le cinquième serveur héberge le composant Storage-as-a-service afin de stocker les fichiers des clients, les arbres WSDL sélectionnés et les services composites , (v) le sixième serveur héberge les comptes des développeurs humains, et stocke les réclamations et les questions des clients , (vi) le dernier serveur héberge le nœud de sauvegarde qui permet de stocker les données et les codes du nœud de vérification et classification.

Type de DC	Serveurs d'hébergement	MIPS	Cout d'utilisation des VM ($)	Fuseau horaire
Premier DC	4	10000	0.3	GMT +6
deuxième DC	9	10000	0.6	GMT +6
troisième DC	7	20000	0.9	GMT +6
IaaS DCs	6	20000	0.6	Selon la région

Tableau VI.1: Caractéristiques des Datacenter

En plus, les emplacements des agents mobiles sont hébergés par les serveurs qui hébergent les agents administrateurs de chaque nœud afin de faciliter la communication entre eux. Enfin, les agents d'infrastructure ont la capacité de s'auto-dupliquer sur tous les serveurs afin de gérer les ressources physiques et virtuelles.

Nous créons également, dans chaque région d'Internet, un Datacenter qui représente les nœuds IaaS, pour tester l'évolutivité des ressources du moteur proposé.

Tout d'abord, nous mettons les Datacenter des principaux nœuds dans la région de l'Asie(R3), puis nous commençons à créer les bases des clients dans les régions: d'Asie (R3), d'Amérique du Nord (R0), d'Amérique du Sud (R1), d'Europe (R2), d'Afrique (R4), et d'Australie (R5), pour évaluer l'impact du nombre des clients sur les performances des systèmes de gestion de chaque nœud.

Client bases number	World Region	Peak Hour	Online Clients During Peak Hours	Online Clients During Off-peak Hours
6	R1/R0/R2/R3/R4/ R5	GMT: 15h-19h /13h-17h /20h-00h	90000 /60000 /40000	40000 /26600 /17700

		/01h-5h /21h-01h /05h-09h	/20000 /10000 /5000	/ 8800 /4400 /2200

Tableau VI.2: Caractéristiques des bases des clients

Les essais durent 4 heures, en supposons que la plupart des clients utilisent le système proposé entre 19h et 23h le soir et que 90% des clients inscrits sont en ligne pendant les heures de pointes, et 40% de clients inscrits sont en ligne pendant les heures hors-pointe. Le nombre des clients qui accèdent simultanément à partir d'une base d'utilisateurs unique est 100000 et le nombre de requêtes simultanées qu'un serveur peut prendre en charge est égal à 10000. Les figures ci-dessous représentent les six bases des clients, chacune représente une région d'Internet dans sa période de pointe.

Le tableau suivant résume les principales taches exécutées par le système proposé :

Nœuds	**Numéro de la tache**	**Description de la tâche**	**Rôle des agents**
Nœud de réception	1	Création des fichiers clients.	Supervisé par la collaboration entre l'AA et l'AMI, et exécuté par le composant Sas.
	2	Gestion des agents mobiles.	Supervisé et exécuté par l'AA.
	3	Gestion du nœud de sauvegarde.	Supervisé par la collaboration entre l'AA et l'AIf
	4	Indexation des réclamations et des questions des clients.	Supervisé par la collaboration entre l'AS et l'AMC, et exécuté par le composant HSaaS.
Nœud de Découverte	5	Stockage des fichiers clients et WSDL.	Supervisé par la collaboration entre l'AA, l'AMC, et l'AIf, et exécuté par le composant StaaS.
	6	Gestion des agents mobiles.	Supervisé et exécuté par l'AA.
	7	Gestion du nœud de sauvegarde.	Supervisé par la collaboration entre l'AA, l'AIf, l'AP et l'AM.
	8	Création des arbres WSDL.	Supervisé par la Collaboration entre l'AA, l'AMD, et l'AMM, et exécuté par le composant DiaaS.

	9	indexation des arbres WSDL	Les taches 9 et 10 sont Supervisées par la Collaboration entre l'AP et l'AIf, et exécutés par le composant DiaaS.
	10	Prétraitement de la requête	
	11	Parcours les arbres WSDL.	De la tache 11 à 13 : les taches sont Supervisées par la Collaboration entre l'AM et l'AIf, et exécutées par le composant DiaaS.
	12	Calculer le dégrée d'existence globale.	
	13	Sélection des services atomiques.	
Nœud de Vérification et classification	14	Stockage des fichiers clients et services sélectionnés.	Supervisé par la collaboration entre l'AA, l'AMC, et l'AIf et exécuté par le composant StaaS.
	15	Gestion des agents mobiles	Supervisé et exécuté par l'AA.
	16	Gestion du nœud de sauvegarde.	Supervisé par la collaboration entre l'AA, l'AIf, l'AQ, ACa et l'AC.
	17	Calculer les scores QoS.	Supervisé par la Collaboration entre l'AQ et l'AIf, et exécuté par le composant CaaS.
	18	Création des tables de catégorisation	Supervisé par la Collaboration entre l'ACa et l'AIf, et exécuté par le composant CaaS.
	19	Exécuter les tests de connectivités	De la tache 18 à 20 : les taches sont Supervisées par la Collaboration entre l'AC et l'IfA, et exécutées par le composant CaaS.
	20	Calculer le cout d'utilisation global.	
	21	Classer les services atomiques et/ou composites.	
		Déploiement des	Supervisé par la Collaboration entre

Nœuds IaaS	22	systèmes de gestion des nœuds.	l'AAs, l'AMR et l'AIf.
	23	Déploiement des composants Cloud.	Supervisé par la Collaboration entre l'AP, l'AM, l'AQ, l'ACa, l'AC, l'AMR et l'AIf (selon le nœud cible).

Tableau VI.3: Résumé des taches exécutées par le système proposé

Enfin, nous utilisons un ensemble de fichiers WSDL recueillis à partir de la base [153] pour valider les algorithmes de découverte et composition de services proposés. Puisque les catégories des fichiers WSDL sur [153] sont déséquilibrés [154], nous choisissons50 fichiers WSDL à partir de quatre catégories : 10 de la catégorie des convertisseurs, 10 de la catégorie des finances, 15 de la catégorie news, et15 de la catégorie des entreprises économiques.

Enfin toutes les expérimentations ont été réalisées sur un ordinateur du type Intel Pentium IV, avec 2.00 GHz de fréquence d'horloge et 4 Go de RAM, sous le système d'exploitation Windows 7 professionnel.

VI.4. Algorithme de référence pour la comparaison

Afin d'évaluer les performances du moteur proposé nous le comparons avec les méthodes de gestion de ressources les plus connues dans le domaine :

1. Méthodes linéaire : avec la méthode linéaire, le système de gestion de ressources choisit, la ressource dont le nœud cible a besoin, parmi les ressources enregistrées sur la Table d'auto-approvisionnement de ressources, et cela en parcourant cette dernière, *ressource après ressource* jusqu'à ce qu'il trouve la ressource qui satisfait ces besoins, et non pas la meilleure ressource qui les satisfait.
2. Méthodes aléatoire : avec la méthode aléatoire, le système de gestion de ressources choisit, la ressource dont le nœud cible a besoin, parmi les ressources enregistrées sur la Table d'auto-approvisionnement de ressources,et cela en parcourant cette dernière, de façon aléatoire jusqu'à ce qu'il atteint la ressource qui satisfait ces besoins, et non pas la meilleure ressource qui les satisfait.

VI.5. Présentation du prototype

Dans cette section nous présentons le prototype qu'on a développé afin de tester le système proposé :

VI.5.1. Architecture globale du prototype

L'architecture présentée par la figure ci-dessous représente la structure du prototype implémenté:

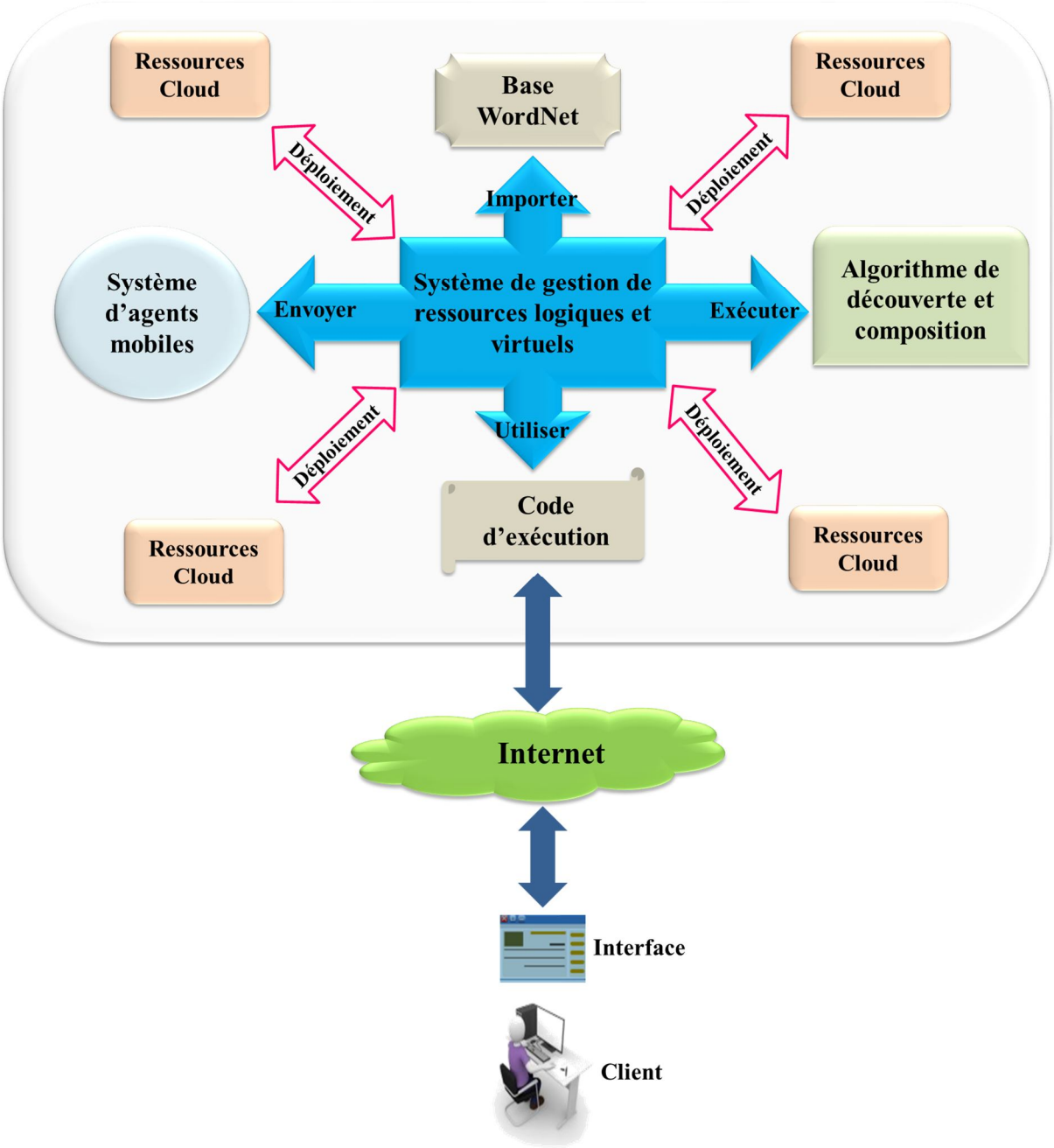

Figure VI.2: L'architecture globale du prototype.

Ce prototype est constitué de trois principales parties : les interfaces utilisateurs, les composants Cloud, et les agents.

VI.5.2. Présentation des Interfaces du système

Dans cette section nous présentons les interfaces qui permettent aux clients d'accéder et d'utiliser le moteur proposé, ces interfaces sont développées avec le langage JSP (JAVA Server Page) :

1. Interface de connexion : cette interface permet aux clients d'entrés dans leurs comptes pour saisir leurs requêtes. Le client saisi son nom d'utilisateur et mot de passe pour accéder à son compte.

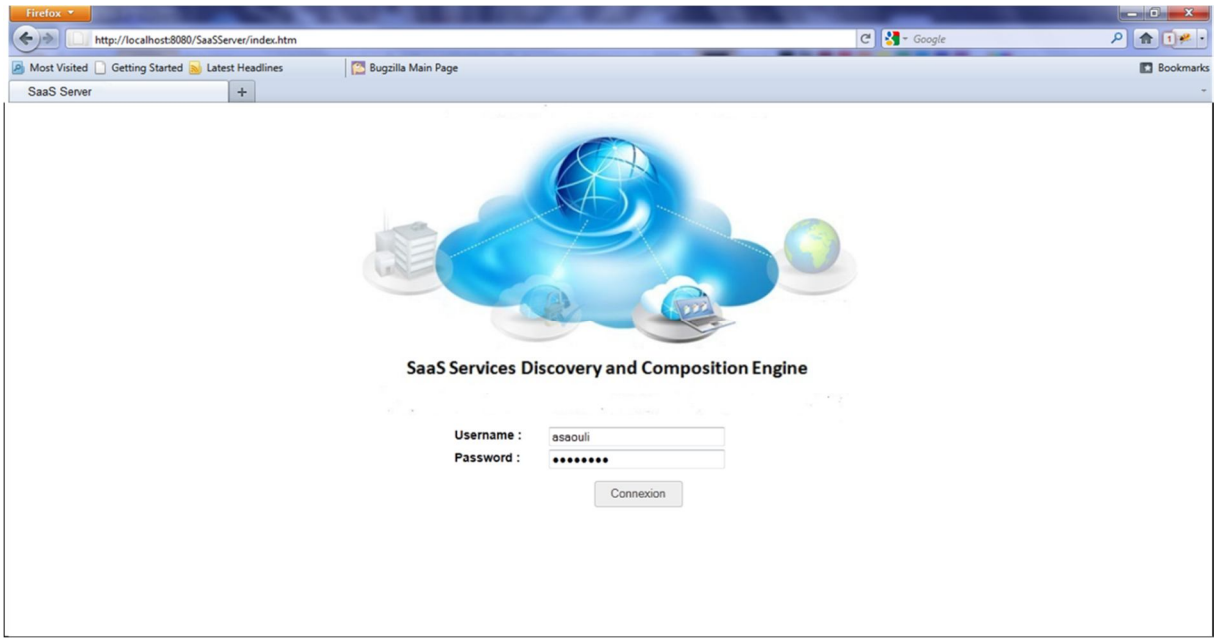

Figure VI.3: L'interface de connexion.

2. Interface de requête : cette interface permet au client de saisir : la requête sous forme textuel (en langage naturelle), les poids de chaque paramètre QoS, et ses choix à propos des paramètres Cloud du service qu'il recherche. Ensuite, le client lance l'opération de découverte et composition et attend les résultats.

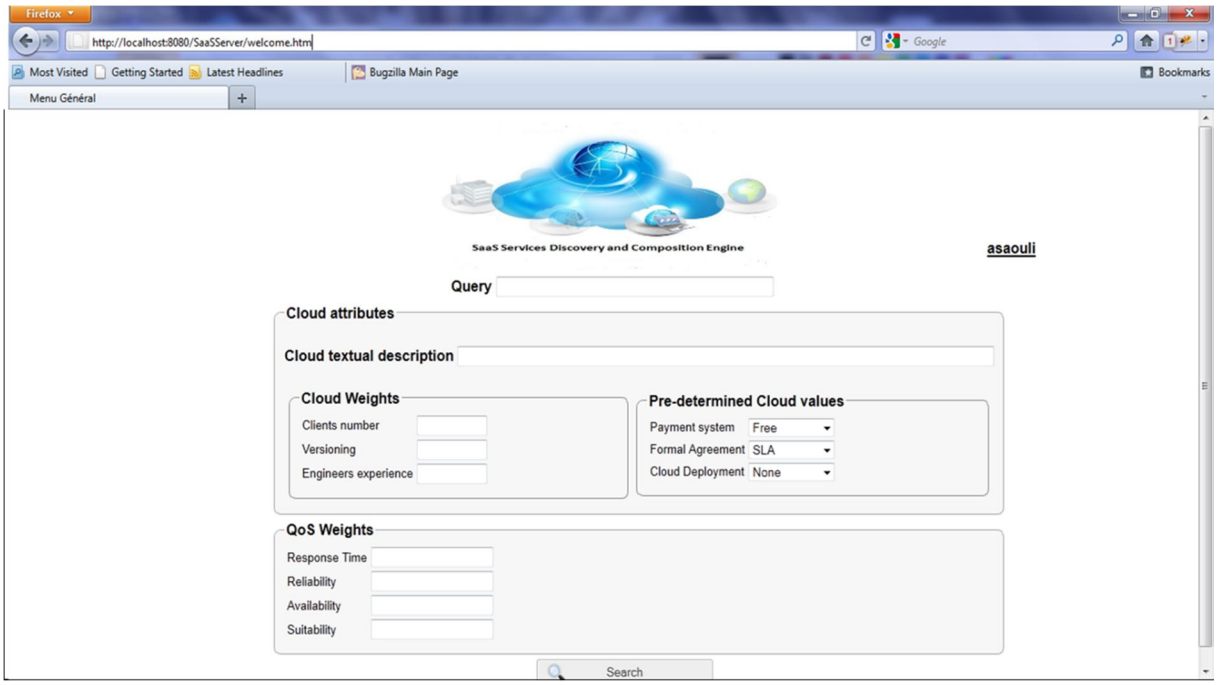

Figure VI.4: L'interface de saisie de la requête.

3. Interface des résultats : c'est l'interface qui permet d'afficher les résultats au client en affichant : les services SaaS, les class d'existence, les degrés d'existence globale, les scores QoS et les similarités Cloud.

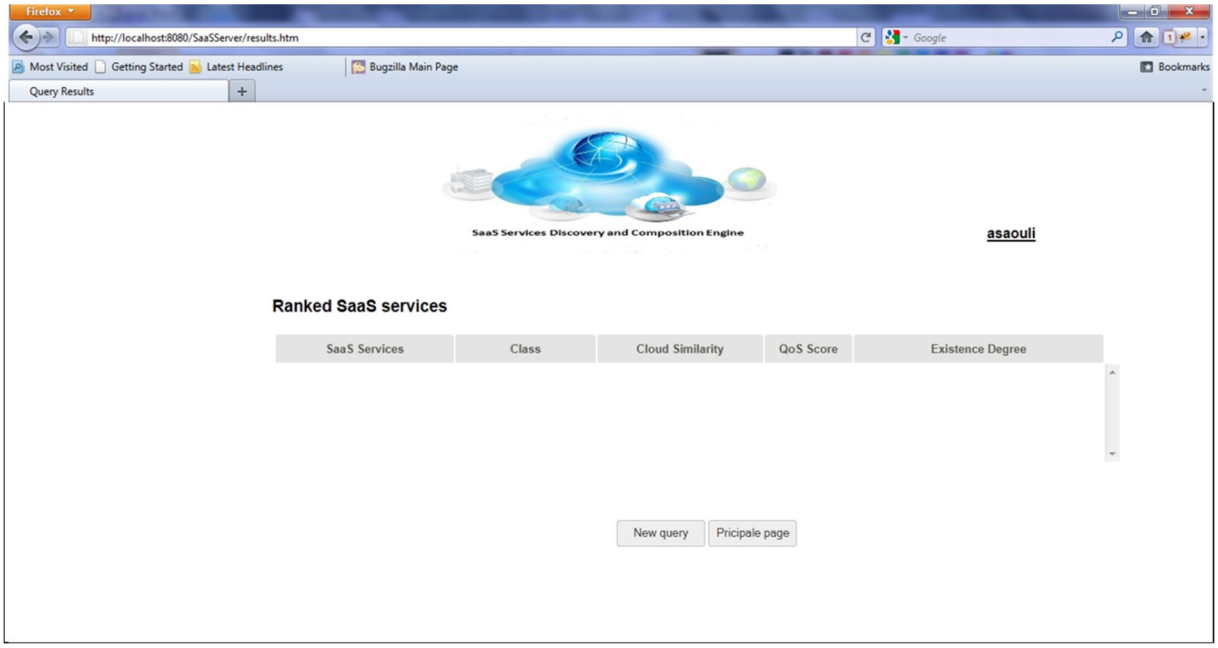

Figure VI.5: L'interface d'affichage des résultats.

4. Interface de contact :

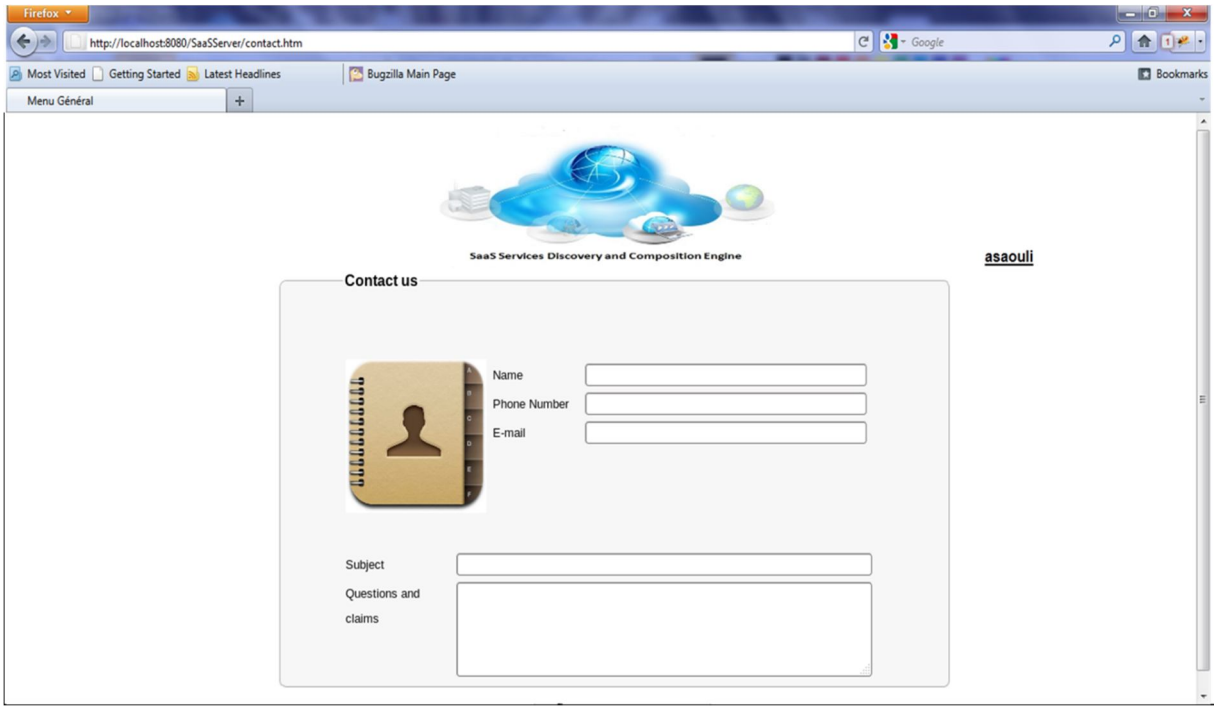

Figure 0.6: L'interface de contact.

VI.5.3. Implémentation d'agents

Dans la plateforme supportant l'exécution du système de découverte et composition proposée, on distingue 2 groupes d'agents : agent de gestion de ressources et agents mobiles :

1. Les agents de gestion sont : Agent administrateur, agent d'infrastructure, agent de composition, agent statistiques, agent de prétraitement, agent de matching, agent de qualité et agent de catégorisation. Dans cette section on ne s'intéresse qu'à présenter les principales lignes du code des trois premiers agents (l'agent de composition suit le même principe que le reste des agents de gestion … ce qui fait que son code ressemble au code des autres agents) vue leur importance pour la gestion de chaque nœud Cloud.
2. Les agents mobile sont : Agent de découverte, agent de ressources, agent interface, Agent de communication, et agent de mis-a-a-jours. Dans cette section on ne s'intéresse qu'à présenter les principales lignes du code des deux premiers agents, vue leur importance pour le bon fonctionnement du système.

La communication entre agent se fait par l'échange de messages, FIPA-ACL en respectant les comportements et algorithmes définis dans les chapitres 4 et 5, par la machine à état fini (Finit State Machine). Enfin, les comportements d'agent sont implémentés en utilisant la class *FSMBehaviour* comme suit :

1. Agent administrateur : le Protocole général d'exécution de l'agent administrateur est comme suit :

```
1. import jade.core.Agent;
2. import jade.core.behaviours.CyclicBehaviour;
3. import jade.core.behaviours.FSMBehaviour;
4. public class Agent_Admin extends Agent{
5. protectedvoidsetup(){
6. // création d'un comportement FSM pour l'agent administrateur

7. FSMBehaviourAgen_Admin = new FSMBehaviour();
8. //definition des etats
9. Agen_Admin.registerFirstState(new Attendre(),"A");
10. Agen_Admin.registerFirstState (new InitiateurNœud(),"B");
11.    Agen_Admin.registerState(new StockFichier(),"C");
12.     Agen_Admin.registerState(new DistributeurTâches(),"D");
13.     Agen_Admin.registerState(new ContratRessources(),"E");
14.     Agen_Admin.registerLastState(new GestionADR(),"F");
15.     Agen_Admin.registerState(new MisàJourNœud(),"G");
16.  //définition des transactions
17.     Agen_Admin.registerDefaultTransition("A","B");
18.     Agen_Admin.registerDefaultTransition("B","C");
19.     Agen_Admin.registerDefaultTransition("C","D");
20.     Agen_Admin.registerTransition("A","B",1);
21.     Agen_Admin.registerTransition("B","A",2);
22.     Agen_Admin.registerTransition("A","C",3);
23.     Agen_Admin.registerTransition("C","A",4);
24.     Agen_Admin.registerTransition("A","D",5);
25.     Agen_Admin.registerTransition("A","B",1);
26.     Agen_Admin.registerTransition("B","E",6);
27.     Agen_Admin.registerTransition("E","F",7);
28.     Agen_Admin.registerTransition("A","B",1);
29.     Agen_Admin.registerTransition("B","G",8);
30.
31.  // appliquer les comportements
```

```
32.  addBehaviour(Agen_Admin);
33.  }
34. }
```

2. Agent d'infrastructure : le Protocole général d'exécution de l'agent infrastructure est comme suit :

```
1. import jade.core.Agent;
2. import jade.core.behaviours.CyclicBehaviour;
3. import jade.core.behaviours.FSMBehaviour
4. import jade.core.behaviours.TickerBehaviour;
5. public class Agent_Infra extends Agent{
6. protectedvoidsetup(){
7. // création d'un comportement FSM pour l'agent Infrastructure

8. FSMBehaviourAgen_Infra = new FSMBehaviour();
9. //definition des etats
10.  Agen_Admin.registerFirstState(new Attendre(),"A");
11.   Agen_Infra.registertState  (new GestionMV(),"B");
12.   Agen_Infra.registerState(new DistributeurMV(),"C");
13.    Agen_Infra.registerLastState(new PrédicteurBesoin(),"D");
14.  //définition des transactions
15.    Agen_Infra.registerDefaultTransition("A","C");
16.    Agen_Infra.registerDefaultTransition("A","D");
17.    Agen_Infra.registerTransition("A","B",1);
18.    Agen_Infra.registerTransition("B","C",2);
19.  // création d'un comportement périodique
20.    addBehaviour(new TickerBehaviour(this, 1000) {
21.    protectedvoidonTick() {
22.    Agen_Infra.registerTransition("A","D",3);
23.    }
24.    }
25.  // appliquer les comportements
26.  addBehaviour(Agen_Infra);
27.  }
28. }
```

3. Agent de composition : le Protocole général d'exécution de l'agent de composition est comme suit :

```
1. import jade.core.Agent;
2. import jade.core.behaviours.CyclicBehaviour;
3. import jade.core.behaviours.FSMBehaviour;
4. import jade.core.behaviours.WakerBehaviour;
5. public class Agent_Comp extends Agent{
6. protectedvoidsetup(){
7. // création d'un comportement FSM pour l'agent de composition

8. FSMBehaviourAgen_Comp = new FSMBehaviour();
9. //definition des etats
10.   Agen_Comp.registerFirstState(new Attendre(),"A");
11.   Agen_Comp.registerState    (new CaaSInitiateur(),"B");
12.   Agen_Comp.registerState(new VérificateurSS(),"C");
13.    Agen_Comp.registerLastState(new ClasseurSS(),"D");
14.  //définition des transactions
15.    Agen_Comp.registerDefaultTransition("A","B");
16.    Agen_Comp.registerDefaultTransition("B","D");
17.    Agen_Comp.registerTransition("A","B",1);
18.    Agen_Comp.registerTransition("B","A",2);
```

```
19.     Agen_Comp.registerTransition("A","C",3);
20.   // création d'un comportement à durée prédéterminée
21.     addBehaviour(new WakerBehaviour(this, 1000)
22.     {
23.     protectedvoidhandleElapsedTimeout() {
24.     Agen_Comp.registerTransition("C","C",4);
25.     Agen_Comp.doDelete();
26.      }
27.     }
28.     Agen_Comp.registerTransition("C","A",5);
29.     Agen_Comp.registerTransition("A","D",6);
30.   // appliquer les comportements
31.   addBehaviour(Agen_Comp);
32.   }
33. }
```

4. Agent mobile de découverte : le Protocole général d'exécution de l'agent mobile de découverte est comme suit :

```
1. import jade.core.Agent;
2. import jade.core.behaviours.OneShotBehaviour;
3. importjade.core.behaviours.TickerBehaviour;
4. import jade.core.behaviours.FSMBehaviour;
5. public class Agent_Comp extends Agent{
6. protectedvoidsetup(){
7. // création d'un comportement FSM pour l'agent de composition

8. FSMBehaviourAgen_Comp = new FSMBehaviour();
9. //definition des etats
10.    Agen_Comp.registerFirstState(new Attendre(),"A");
11.    Agen_Comp.registerState    (new ChercheurSS(),"B");
12.    Agen_Comp.registerLastState(new DistributeurAD (),"C");
13.   //définition des transactions
14.     Agen_Comp.registerDefaultTransition("A","A");
15.     Agen_Comp.registerTransition("A","C",1);
16.     Agen_Comp.registerTransition("C","B",2);
17.   //création d'un comportement périodique
18.     addBehaviour(new TickerBehaviour(this, 1000) {
19.     protectedvoidonTick() {
20.     Agen_Comp.registerTransition("B","B",3);
21.     Agen_Comp.registerTransition("B","C",4);
22.       }
23.     }
24.   // appliquer les comportements
25.   addBehaviour(Agen_Comp);
26.   }
27. }
```

28. Agent mobile ressources : le Protocole général d'exécution de l'agent mobile de ressources est comme suit :

```
1. import jade.core.Agent;
2. import jade.core.behaviours.OneShotBehaviour;
3. importjade.core.behaviours.TickerBehaviour;
4. import jade.core.behaviours.FSMBehaviour;
5. public class Agent_Comp extends Agent{
6. protectedvoidsetup(){
7. // création d'un comportement FSM pour l'agent de composition
```

```
8. FSMBehaviourAgen_Comp = new FSMBehaviour();
9. //definiton des etats
10.    Agen_Comp.registerFirstState(new Attendre(),"A");
11.    Agen_Comp.registerState    (new ChercheurSS(),"B");
12.    Agen_Comp.registerLastState(new DistributeurAD (),"C");
13.    Agen_Comp.registerLastState(new DéployerComp(),"D");
14.  //définition des transactions
15.     Agen_Comp.registerDefaultTransition("A","A");
16.     Agen_Comp.registerTransition("A","C",1);
17.     Agen_Comp.registerTransition("C","B",2);
18.  //création d'un comportement périodique
19.     addBehaviour(new TickerBehaviour(this, 1000) {
20.     protectedvoidonTick() {
21.     Agen_Comp.registerTransition("B","B",3);
22.     Agen_Comp.registerTransition("B","C",4);
23.       }
24.     }
25.     Agen_Comp.registerTransition("B","A",5);
26.     Agen_Comp.registerTransition("A","D",6);
27.  // appliquer les comportements
28.  addBehaviour(Agen_Comp);
29.  }
30. }
```

Le constructeur *Attendre()* est définis dans une classe héritière de la classeOneShotBehaviour, qui représente un comportement simple, et qui a une signification distincte pour chaque type d'agent : l'agent administrateur attend la venue d'une nouvelle requête client, l'agent infrastructure attend une nouvelle requête pour l'obtention de nouvelles ressources. L'agent composition attend les services sélectionnés, enfin les agents mobiles attendent les directives des agents administrateurs pour les dispatchers sur Internet. Le code de cette procédure est comme suit :

```
1. Private class Attendre OneShotBehaviour {
2. Public voidaction() {
3. block();// l'agent est bloqué en attente d'un méssage
4. }
5. }
```

Les messages échangés entre agents sont basés sur la spécification FIPA-ACL. Le code suivant illustre la réception et l'envoi de locution via un message FIPA-ACL entre l'agent administrateur qui demande une machine virtuelle et l'agent infrastructure qui lui fournit une MV.

L'agent administrateur commence à envoyer un message à l'agent infrastructure comme suit :

```
1.  import jade.core.AID;
2.  import jade.core.Agent;
3.  import jade.core.behaviours.FSMBehaviour;
4.  import jade.core.behaviours.OneShotBehaviour;
5.  import jade.lang.acl.ACLMessage;
6.  public class Agent_Admin extends Agent {
7.  protected void setup(){
8.  FSMBehaviourAgent_Admin = new FSMBehaviour();
9.   Agent_Admin.registerFirstState(new          attendreAgentInfra(),
     "attendreAgentInfra");
10. Agent_Admin.registerState(new envoiDemandMV(), "envoiDemandMV");
```

```
11. Agent_Admin.registerLastState(new fin(), "fin");
12. /*****************************************************************/
13. Agent_Admin.registerDefaultTransition("attendreAgentInfra",
    "envoiDemandMV");
14. Agent_Admin.registerTransition("envoiDemandMV","attendreAgentInfra"
    ,1);
15. Agent_Admin.registerTransition("envoiDemandMV", "fin", 2);
16. addBehaviour(agentA_Admin);
17. }
18. /*****************************************************************/
19. private class attendreAgentInfra extends OneShotBehaviour{
20. public voidaction() {
21. block();
22.  }
23. }
24. /*****************************************************************/
25. private class envoiDemandMV extends OneShotBehaviour{
26. public voidaction() {

27. ACLMessage message = new ACLMessage(ACLMessage.INFORM);
28. message.addReceiver(new AID("Agent_infra", AID.ISLOCALNAME));
1.  message.setContent("Demande Nouvel MV");
2.  send(message);
3.  }

4.  /*****************************************************************/
5.  private class fin extends OneShotBehaviour{
6.  public voidaction() {
7.  myAgent.doDelete();
8.   }
9.  }
10. }
```

L'agent d'infrastructure reçoit le message envoyé par l'agent administrateur comme suit:

```
1.  import java.security.acl.Acl;
2.  import jade.core.AID;
3.  import jade.core.Agent;
4.  import jade.core.behaviours.FSMBehaviour;
5.  import jade.core.behaviours.OneShotBehaviour;
6.  import jade.lang.acl.ACLMessage;
7.  public class Agent_Infra extends Agent {
8.  protected void setup(){
9.  FSMBehaviouragent_Infra= new FSMBehaviour();
10.  agent_Infra.registerFirstState(new attendreDemandMV(),
     "attendreDemandMV");
11. agent_Infra.registerState(new MV(), "MV");
12. agent_Infra.registerLastState(new fin(), "fin");
13. /*****************************************************************/
14. agent_Infra.registerDefaultTransition("MV", "attendreDemandMV");
15. agent_Infra.registerTransition("attendreDemandMV", "MV",1);
16. agent_Infra.registerTransition("attendreDemandMV", "fin",2);
17. addBehaviour(agent_Infra);
18. }
19. private class attendreDemandMV extends OneShotBehaviour{
20. public voidaction() {
21. block();
22.
23. }
```

```
24. /*************************************************************/
25. private class VM extends OneShotBehaviour{
26. public voidaction() {
27. ACLMessagemessageRecu = receive();
28. ACLMessage message = new ACLMessage(ACLMessage.INFORM);
29. message.addReceiver(new AID("Agent_Admin", AID.ISLOCALNAME));
30. message.setContent("MV ID:002");
31. send(message);
32. }
33. }
34. /*************************************************************/
35. private class fin extends OneShotBehaviour{
36. public voidaction() {
37. myAgent.doDelete();
38.  }
39. }
40. }
```

VI.6. Description du Modèle d'évaluation des performances

Pour évaluer le système proposé, nous utilisons 2 mesures qui nous permettent decomparer entre les mécanismes proposés, avec les méthodes linéaires et aléatoires:

- Temps de réponse moyen : il représente le temps de traitement des requêtes des clients:

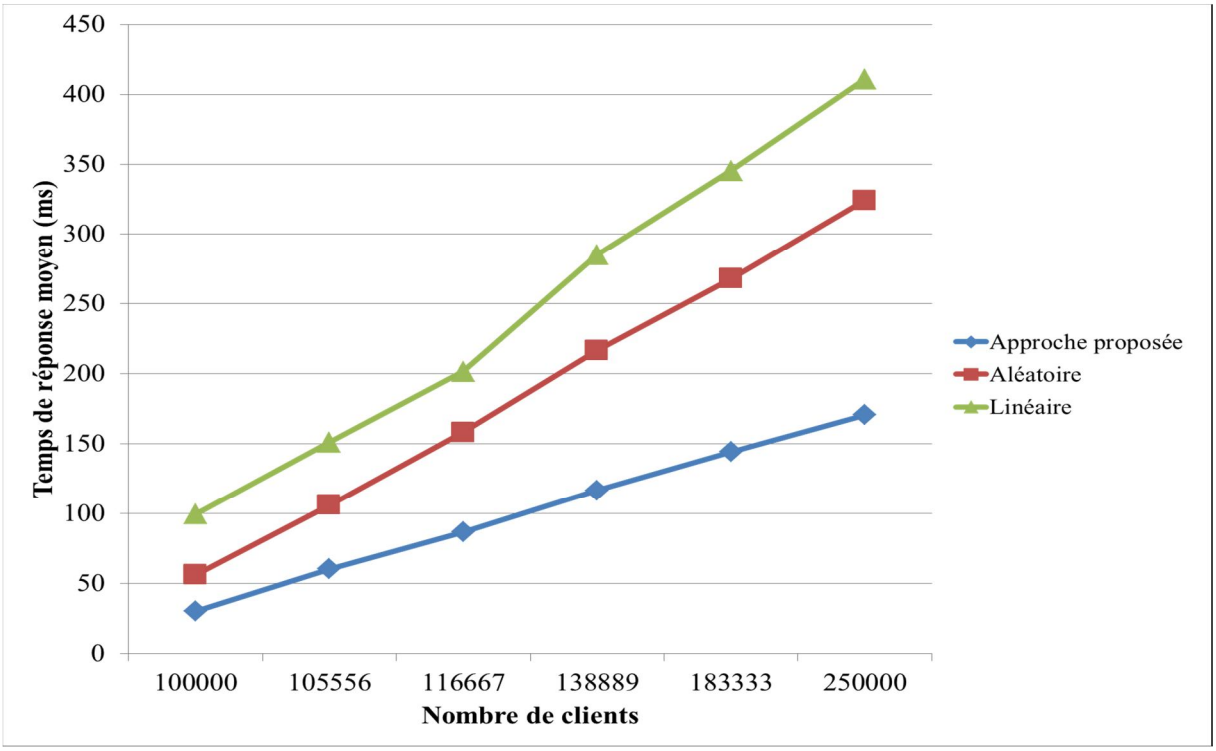

Figure VI.7: Résumé des résultats du temps de réponse moyen.

- Coût d'utilisation(CU): le coût est calculé sur la base du nombre des machines virtuelles utilisées pour traiter les requêtes des clients:

$$CU = \sum_{j-1}^{NUD}((NUVM_j * CVM_j) + CTD_j) \quad (23)$$

Où :

NUD: représente le nombre de Datacenter utilisés.

CVM_j : représente le cout d'utilisation d'une machine virtuelle.

$NUVM_j$: représente le nombre de machines virtuelles utilisées dans le Datacenter *j*.

CTD_j : représente le coût de transfère de donnée sur le Datacenter *j*.

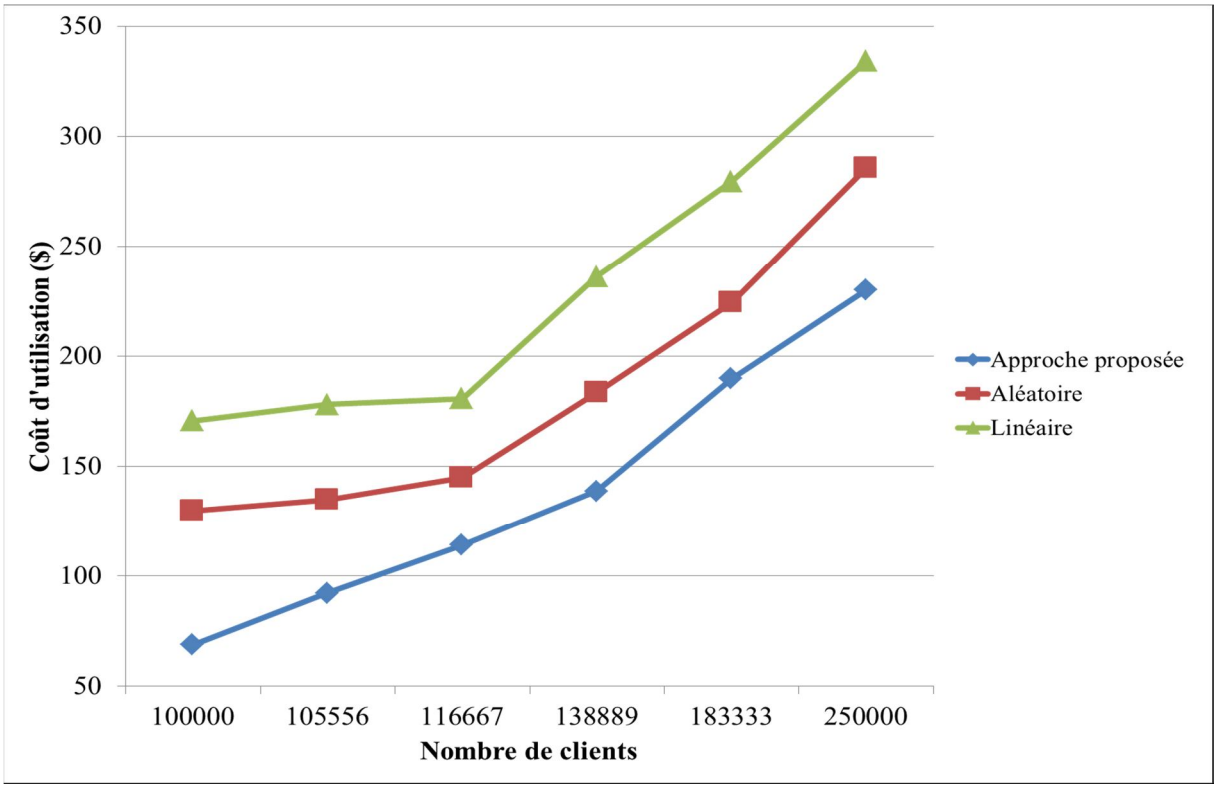

Figure VI.8: Résumé des coûts d'utilisation moyenne.

VI.6. Discussion et Etude comparative

Dans cette section nous comparons l'approche que nous proposons avec celles présentées sur les figures de comparaison présentée ci-dessus, nous discutons les raisons pour lesquelles le mécanisme qu'on propose représente des résultats meilleurs en général:

Les figures VI.6 et VI.7 montrent les performances du système proposé par rapport aux méthodes : linéaire et aléatoire. De ces résultats, nous déduisons que :

1. Le système de gestion du moteur réduit le temps de réponse (Figure VI.6) par rapport aux méthodes aléatoires, et linéaires, en raison de la gestion draconienne des machines virtuelles et CPU, mais notamment, due à la collaboration entre les différents agents de supervision de ressources, ce qui réduit le temps de latence pour l'attribution des ressources dans chaque nœud Cloud.

2. Enfin, on remarque que le système de gestion proposé (Figure VI.7) réduit les coûts par comparaison aux méthodes aléatoires et linéaires qui utilisent un grand nombre de machines virtuelles. Cependant, nous continuons à penser que les coûts sont très élevés à cause du modèle économique utilisé par le simulateur Cloudsim, qui est basé sur le nombre de machines virtuelles utilisées. Nous pensons que ce modèle n'est pas bien adapté à la nature du moteur proposé ou tout type de moteur qui soit destiné à la recherche d'informations, puisque l'AIf peut utiliser la même VM pour traiter plusieurs requêtes au même temps. Donc, de façon indirecte, plusieurs clients peuvent utiliser la même machine virtuelle en même temps, alors pourquoi le client paye pour une machine virtuelle qu'il ne possède pas seul?! Nous pensons aussi que la meilleure façon de calculer les coûts d'utilisation doit être basée sur le temps pendant lequel la requête du client occupe les ressources du système.

VI.7. Conclusion

Dans ce chapitre nous avons vu les outils utilisés pour l'implémentation du prototype, ainsi que la base de fichiers WSDL utilisés pour valider les algorithmes de matching. L'architecture du prototype illustre les composants implémentés ainsi que leurs relations. Enfin nous avons présenté quelques codes JAVA qui illustrent l'implémentation des comportements d'agent, et la communication entre eux.

Les expérimentations faites sur le prototype nous ont permis de valider l'approche proposée dans cette étude.

Ainsi le prochain chapitre présente un nouveau modèle de comparaison entre services SaaS qui nous permet de mener une étude comparatives entre l'aspect matching de l'approche proposé et les algorithmes de matching présentés dans le chapitre 3 (Travaux liées à l'approche proposée), d'un côté, et d'un autre côté, entre l'aspects architectural de l'approche proposée et les architectures basées découverte et composition présentées dans le chapitre 3 également.

Partie III. Etude Comparative

Chapitre VII : Comparaison et Evaluation.

Pourquoi la comparaison? Quel modèle adopté pour la comparaison? Quelles sont les principales catégories distinctes entre approches de découverte et composition? Comment classer les mécanismes de découverte et composition? Quelles sont les catégories des critères adoptés pour la comparaison? Quels sont les critères d'évaluation adoptés et proposés? Quelles sont les principales limitations des travaux connexes? Quelles sont les principales différences et similarités entre l'approche proposée et les travaux connexes?

VII
Comparaison et Evaluation

VII.1. Introduction

Dans ce chapitre nous focalisons sur la proposition d'un nouveau modèle de comparaison entre approches de découverte et composition de services SaaS. Le modèle de comparaison proposé prend en considération les trois aspects des approches de découverte et composition de services : aspect matching, aspect architectural, et aspect évaluative.

L'importance d'un tel modèle, réside dans le besoin de catégoriser le grand nombre d'approches de découverte et composition afin de mieux les analyser et les comparer les unes par rapports aux autres, afin de pouvoir extraire les avantages et inconvénients de chaque approche pour développer de meilleurs mécanismes.

Nous nous basons également sur les modèles présentés dans [155, 156, 157], pour proposer notre modèle de comparaison. . Il y a en effet, une grande similitude entre services SaaS et services web. . Nous allons tenter de répertorier les apports de ces modèles et porter un regard critique sur les approches de découverte et composition de services.

Ce chapitre est organisé comme suit :
Dans la section 2 nous présentons la taxonomie que nous proposons pour catégoriser les services SaaS. Dans la section 3, nous présentons les différentes classes des approches de découverte et composition. Dans la section 4, nous présentons en détail les critères d'évaluation. Dans la section 5, nous rappelons brièvement les approches de découverte et composition de services présentées dans le chapitre 3 et nous les comparons avec l'approche que nous proposons dans cette thèse. Enfin, la section 6 est la conclusion de ce travail.

VII.2. Taxonomie

Afin d'avoir une vue, à la fois globale et profonde des systèmes de découvertes de services dans le Cloud, nous proposons une taxonomie basée sur sept points de vue différents : vue modélisation, vue distributionnelle, vue traitement, vue d'automatisation, vue service, vue matching, et vue hybride. La figure ci-dessous illustre cette taxonomie:

VII.2.1. Vue modélisation

Ce point de vue concerne la manière dont les systèmes de découverte modélisent leurs principaux composants. En utilisant cette vue, les systèmes de découverte peuvent être catégorisés en systèmes basés agent ou basés objet. Les systèmes basés agent sont dotés d'un comportement social faisant émerger un raisonnement intelligent qui vise à optimiser les résultats obtenus par le système. Toutefois, le paradigme agent n'est pas encore aussi stable que celui basé objet, vue que l'émergence de nouveaux comportements basés agent n'est pas assez contrôler et assez orienter vers l'optimisation des résultats.

VII.2.2. Vue distribution

Ce point de vue concerne la manière dont les systèmes de découverte, distribuent et stockent les descriptions des services sur le réseau. On distingue trois types de distribution :

(1) Centralisé : qui désigne l'utilisation d'un registre global central pour stocker les descriptions des services. Ce registre est contrôlé par un organisme neutre.

(2) Index : désigne l'utilisation d'un ensemble d'index distribués sur Internet pour stocker, chacun, un groupe de services.

(3) Décentralisé : qui désigne l'utilisation de descriptions contenues et publiées sur la plateforme de chaque fournisseur. De façon générale ici, on on utilise la technologie Peer-to-Peer pour relier les descriptions des services.

VII.2.3. Vue traitement

Ce point de vue désigne la technologie de traitement d'informations (stockage, distribution de CPU, gestion de RAM) associée au Cloud, que les systèmes de découverte utilisent pour concevoir leurs architectures. On distingue trois principaux types de technologie : Grid, Green et Système classique.

VII.2.4. Vue d'automatisation

Ce point de vue concerne la manière dont les systèmes de découverte sont utilisés. En effet, les systèmes de découverte peuvent être soit manuels soit automatiques [158, 159]. Un système de découverte manuel est appelé par un utilisateur humain au moment de la conception pour la recherche de services. Quand il est automatique, il est appelé par un agent système, soit au moment de la conception soit au moment de l'exécution pour la recherche des services [155].

VII.2.5. Vue service

Ce point de vue concerne le modèle de livraison que les systèmes de découverte visent à sélectionner. On distingue deux principaux types :

(1) logiciel : il s'agit d'utiliser les aspects fonctionnels, non-fonctionnels, industriels et/ou économiques des logiciels offerts comme services pour découvrir et sélectionner les meilleurs d'entre eux,

(2) Infrastructure: il s'agit d'utiliser la description de l'infrasctructure des services (capacité en espace de stockage, nombre de CPU … etc.) pour découvrir et sélectionner les meilleurs d'entre eux.

VII.2.6. Vue matching

Ce point de vue concerne l'algorithme de matching utilisé pour déterminer le degré d'adaptation de la requête par rapport à la description du service. On distingue deux principaux types de matching :

(1) Le Matching syntaxique : l'avantage des méthodes syntaxiques est la rapidité du filtrage de grands nombres de serviceset l'absence d'annotations. La conséquence en est l'ambiguïté linguistique n'est pas prise en considération par ces approches. Cela empêche alors la récupération de services décrits différemment, mais offrant les mêmes fonctionnalités. Ce type d'approche ne facilite pas la création de méthodes

automatiques qui prennent en compte la composition, l'invocation et la supervision de services à cause de l'absence de l'aspect sémantique [160].

(2) Le Matching sémantique : ce type d'approches utilise des annotations sémantiques et des ontologies pour calculer le degré de similarité entre la requête et la description du service. Ces approches sont destinées à affiner et améliorerles résultats obtenus par les approches syntaxiques. Cependant, les algorithmes de matching sous jacents sont en général plus complexes et sont plus gourmands en temps d'exécution [160].

VII.2.7. Vue hybride

Ce point de vue concerne la manière dont les techniques et technologies de chaque vue, (précédemment citées), sont combinées par les systèmes de découverte pour accroitre l'efficacité. Par exemple, on peut trouver des systèmes qui utilisent un double filtre : syntaxique pour réduire l'espace de recherche et sémantique pour améliorer la précision.

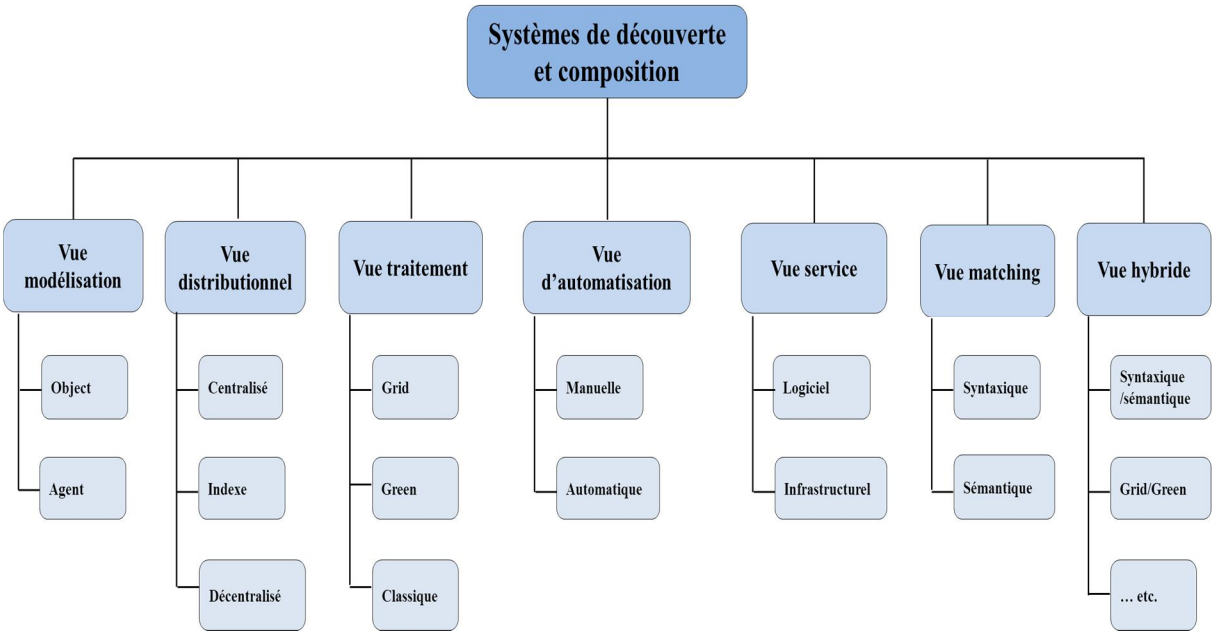

Figure VII.1: Taxonomie des approches de découverte et composition de services SaaS.

VII.3. Classification d'approches

Nous adoptons le modèle présenté dans [155] qui classe les approches de découverte de services en 4 catégories :

VII.3.1. Approche Logique (AL) :

Cette catégorie exploite les standards d'inférences logiques Basées sur des mesures de similarités logiques et des annotations sémantiques des services, afin de déterminer les relations sémantiques entre services. Les approches basées logiques sont plus précise que les approches basées syntaxe, en raison de leur solide base mathématique.

VII.3.2. Approche Non logique (ANL) :

L'Utilisation de la logique formelle pour la découverte des services peut être difficile et complexe. En outre, l'utilisation des approches basées logiques pour le calcul de similarités, peut conduire à une consommation de temps accrue [161]. La découverte de services basés non-Logique vise à surmonter les inconvénients des approches Logiques. Cette catégorie vise à exploiter les relations sémantiques implicites et les fréquences des termes dans les descriptions de services en s'appuyant sur des techniques, telles que les graphes de matching, statistique linguistique, ou extraction d'informations.

VII.3.3 Approche logique et non logique (ALNL):

Les approches basées logiques utilisent une description Sémantique basée sur l'ontologie du domaine, ce qui peut conduire à l'élimination de services qui répondent aux exigences du client. Afin de surmonter cette limitation on combine approche logique et non-logique. L'idée principale est que quand la partie basée logique échoue, la partie basée non-logique peut récupérer les services ignorés [160].

VII.3.4.Approche logique et Syntaxe (ALS):

Cette catégorie combine syntaxe et logique pour la réalisation d'algorithmes de matching.

VII.4. Critères d'évaluation

Dans cette section nous présentons les différents critères permettant la comparaison de l'approche proposée avec celles présentées dans l'état de l'art :

VII.4.1. Critères de matching

- Le support du Clustering: avec l'évolution exponentielle en nombre de services dans le Cloud, il est devenu nécessaire d'avoir un système de clustering (généralement syntaxique) qui permet de réduire l'espace de recherche de services, (nombre de services Cloud sur lesquels l'algorithme de recherche sémantique va être appliqué). Cela permet de réduire le temps de réponse; qui sera, autrement, irréaliste si l'algorithme sémantique est directement appliqué sur la base de services Cloud. La présence d'un module de clustering indique que l'algorithme ou l'architecture proposée prend en considération la réduction du temps de réponse.
- Elément de matching : un meilleur algorithme de découverte et composition de services dans le Cloud est un algorithme qui prend en considération tous les aspects et parties descriptives des services Cloud on en compte six parties :
  1. Le domaine : il s'agit de décrire la fonctionnalité globale du service ainsi que sa localisation.
  2. l'aspect fonctionnel: cela permet de décrire ses entrées/sortie ainsi que ses opérations.

3. l'aspect non fonctionnel : cela permet de décrire le niveau de qualité du service offert comme par exemple : le temps de réponse, la disponibilité, la transparence … etc.
4. l'aspect logiciel : cela permet de décrire l'aspect industriel ainsi que le modèle économique définit par le fournisseur du service Cloud.
5. l'aspect infrastructure : cela permet de décrire l'infrastructure sur laquelle le service est déployé, par exemple : le nom du système d'exploitation, nombre d'instances disponibles, nombre de machines virtuelles … etc.
6. les pré-conditions et post-conditions : ce qui permet de décrire les conditions d'accès aux services ainsi que les effets du service sur son environnement.

- Combinaison d'éléments de matching : de façon générale un algorithme de matching passe par quatre principales étapes : découverte, sélection, composition et tri de services. Il serait préférable pour de tels algorithmes de combiner entre tous (ou une partie) les éléments de matching pour chaque étape, afin de s'assurer de la prise en considération de toutes les exigences du client à chaque étape (l'algorithme qu'on propose dans cette thèse met en œuvre partiellement cette idée).
- Support du calcul du degré de matching : Parfois, les descriptions et la requête sont matchées partiellement (partiellement compatibles). Ce qui conduit à ne pas retourner des services qui pourtant répondent à la requête du client. Un meilleur algorithme de découverte et composition est un algorithme qui permet de calculer le degré de similarité entre description et requête [160]. L'importance du degré de matching réside dans le fait qu'il permet de trier les services, retournés aux clients, afin que ces services puissent être choisis plus efficacement, au lieu d'un choix aléatoire.
- Création de classes : afin d'avoir plus de précision et pour offrir plus de confort aux clients dans leurs choix du service à utiliser, il est préférable de catégoriser les services sélectionnés en plusieurs classes, cela indique également que l'algorithme de matching permet d'offrir des résultats plus pertinents.
- Matching multi-étapes : cela permet d'indiquer que le système de découverte effectue le processus de matching sur plusieurs étapes séquentielles ou exécutées en compétitions (parallèle). Enfin, les résultats retournés par chaque étape sont ensuite regroupés pour avoir des résultats plus raffinés [155]. Toutefois, l'utilisation du matching multi-étapes, peut être très gourmande en temps de calcul malgré la bonne qualité des résultats retournés. Un compromis entre qualité et temps de réponse est parfois nécessaire pour améliorer le fonctionnement global du système. Ceci a poussé plusieurs chercheurs à développer des méthodes qui offrent au client le choix entre ces deux critères [162].
- Support de WSDL : de nous jours l'utilisation des fichiers WSDL notamment pour la création d'indexes de services est très répandue. Les descriptions WSDL sont très répandues pour leur faculté à prendre en considération l'aspect syntaxique des services Cloud. Le support des WSDL indique que l'approche proposée peut combiner entre recherche syntaxique et celle sémantique en associant les WSDL à l'algorithme de matching.
- Support de la composition/sélection : avec la complexité des requêtes des clients, il est devenu très dur de trouver des services atomiques simples qui peuvent satisfaire toutes les exigences exprimées dans la requête. D'un autre côté, avant de passer à

l'étape de composition, il est nécessaire de sélectionner les meilleurs services atomiques qui répondent partiellement à la requête afin de créer des services composites plus complexes. Le support de la sélection ainsi que de la composition de services par un système de matching indique que ce dernier optimise la réutilisation ainsi que l'invocation des services retournés aux clients [157].

- Réduction des tests de connectivités : la prise en considération de la composition des services après la sélection des meilleurs services atomiques, n'est pas suffisante par rapport à l'optimisation du temps de réponse ; surtout quand il s'agit d'une requête très complexe qui nécessite la sélection d'un grand nombre de services atomiques. Un algorithme de découverte et composition qui utilise un ensemble de règles ou un algorithme révolutionnaire tel que les algorithmes génétiques (par exemple) pour réduire le nombre de tests de connectivités , en est un exemple.
- Support de WordNet : la base de données lexicale WordNet est considérée comme une ontologie générale qui relie sémantiquement les concepts et termes de la langue anglaise (il existe d'autre version pour WordNet pour les langues espagnole et française aussi). Vu que les services Cloud sont de nature autonome, hétérogène et faiblement couplée, il est nécessaire qu'ils prennent en considération tout type d'ontologie spécifique afin de les relier facilement et efficacement. L'algorithme qui prend en considération l'ontologie générale WordNet peut être étendu pour prendre en considération n'importe quel type d'ontologies spécifiques.
- Exactitude : nous définissons ce critère qualitatif comme la capacité du système de découverte et composition a capturer toutes les exigences du client . Ce critère se base lui-même sur les sous critères suivants :
  1. Elément de matching : plus l'approche prend en considération plus d'éléments descriptifs plus elle est exacte. Pour chaque élément de matching pris en considération un score est assigné à l'approche.
  2. Combinaison d'éléments de matching : plus l'approche utilise plus d'éléments de matching sur chaque étape plus elle est exacte. Le score est égale au nombre d'éléments de matching pris en considération sur chaque étape.
  3. Support du calcul du degré de matching : une approche qui permet de calculer le degré de matching est plus exacte qu'une approche qui ne le permet pas. On accorde un score pour l'approche qui prend en considération le degré de matching.
  4. Création de classes : plus l'approche fournit plus de classes pour catégoriser les services Cloud, plus elle est exacte. On accorde un score pour l'approche qui prend en considération la création de classes.
  5. Matching multi-étapes: si l'approche prend en considération plusieurs étapes de matching alors elle est plus exacte. On accorde un score à l'approche qui prend en considération le matching multi-étapes.
  6. Support de la composition/sélection: si l'approche prend en considération la composition et la sélection en même temps alors elle est plus exacte que ceux qui ne prennent qu'une seule. On attribue un score à l'approche qui prend en considération la composition et la sélection en même temps.
- Evolutivité : nous définissons ce critère qualitatif comme la capacité du système de découverte à s'intégrer avec d'autres systèmes du même genre pour améliorer les

résultats retournés au client. Afin d'estimer l'évolutivité nous utilisons les critères suivants :

1. Support de WordNet : si l'approche proposée peut supporter l'ontologie générale WordNet, pour le service demandeur comme pour les services fournisseurs alors elle est plus évolutive que les approches qui ne le supportent pas.
2. Support de WSDL : actuellement la majorité des services web (qui représentent une interface pour les services Cloud) sont dépourvus de toute annotation sémantique. Ce qui implique que l'approche qui prend en considération les WSDL (en tant qu'outil de description syntaxique) est plus évolutive que celle qui ne prend pas cela en considération.
3. Support del'approche l'hybride : c'est-à-dire que l'approche de découverte et composition peut être combinée avec d'autres approches afin de raffiner les résultats. Par exemple, si une approche s'avère efficace lors de la sélection et pas efficace lors de la composition, et une autre approche qui s'avère efficace lors de la composition, alors si la première approche supporte le remplacement de son mécanisme de composition par celui de la deuxième approche alors elle est plus évolutive que les approches qui n'acceptent pas le remplacement.

Afin d'évaluer le degré d'évolutivité nous nous basons sur ce qui suit : si l'approche supporte les trois critères alors elle est «hautement évolutive», si l'approche supporte deux critères alors elle est «moyennement évolutive», si l'approche supporte un seul critère alors elle est «faiblement évolutive», si l'approche ne supporte aucun critère elle «n'est pas évolutive».

VII.4.2. Critères architectural

- Supporter le recueil automatique de services : cela indique que l'architecture de découverte de services prend en considération le recueil de services, par le biais d'un outil (de type crawler) qui peut voyager sur Internet pour collecter les services Cloud, afin de les indexer sur la plateforme qui supporte l'architecture proposée. Un exemple de tels outils : les agents mobiles.
- Supporter la publication de services : cela indique que l'architecture de découverte et composition offre aux fournisseurs de services Cloud, la possibilité de publier leurs services auprès du moteur de recherche afin qu'il soit indexé. Cela permet de couvrir les services Cloud qui n'ont pas été collectés automatiquement.
- Gestion de ressources : Nous définissons ce critère qualitatif comme capacité du moteur de découverte et composition à gérer ses propres ressources. Les contraintes prises en considération pour estimer la gestion des ressources sont : gestion de machine virtuelle, gestion d'espace de stockage, gestion des distances géographiques, gestion globale de ressources, prédiction de besoin, liaisons de ressources distribuées sur le réseau et la gestion des instances d'applications Cloud qui exécutent les algorithmes de matching. Enfin, un score est attribué à chaque contrainte prise en considération.
- Modélisation du système : Nous définissons ce critère qualitatif comme la capacité du système de découverte et composition à présenter chaque tâche principale par un module (ou agent). La décomposition de l'architecture en plusieurs modules et sous

modules permet de créer un système hautement robuste et tolérant aux fautes. Enfin, on attribue un score à chaque principale tâche présentée par un module ou un agent.

- Index de description : cela indique que l'architecture proposée inclut un index de description de services Cloud, ce qui permet à l'algorithme de matching d'accéder rapidement aux services afin de les filtrer, et cela au lieu d'utiliser un registre centralisé (publié sur le net) pour recueillir les services à chaque fois qu'une requête est reçue. La présence d'un index indique que l'architecture proposée prend en considération la réduction du temps de réponse.
- Robustesse : un moteur de découverte et composition de service devrait être capable de faire face aux erreurs et changement qui surviennent au réseau ainsi qu'au sein de ces propres ressources, sans influencer le bon fonctionnement du système global. La présence d'un système de récupération d'erreur ou de ré-stabilisation du système indique que le moteur de découverte est robuste i.e. en général on utilise un site de sauvegarde (backup) afin de s'adapter au changement et erreurs sur le réseau [157].
- Mise à jour d'index : l'un des problèmes majeurs des services Cloud (comme pour les services web) est le changement des fonctionnalités ainsi que de la qualité de ces derniers par le fournisseur à chaque fois qu'il désire cela. Ce qui implique la nécessité de mettre à jour l'index de description pendant chaque période de temps. Un moteur de découverte doit tenir en compte et vérifier la disponibilité et les changements éventuels des services indexés [157].
- Remplacement de services : la présence d'un module ou d'un agent responsable pour remplacer les services Cloud (retourné au client) en état d'échec, est un élément très important pour augmenter l'efficacité du moteur de découverte [157].
- Support d'un modèle de payement : depuis l'apparition des technologies Grid et Cloud, il est devenu essentiel d'associer à chaque service un modèle économique qui explique le mode de payement pour les clients désirant utiliser le service. La présence d'un modèle de payement pour un moteur de découverte (qui représente lui-même un service) indique que ce dernier prend en considération l'aspect économique du service de découverte et composition.

VII.4.3. Critères de test et réalisation

- Performance: Nous définissons ce critère qualitatif comme la capacité du système de découverte et composition à fournir un ensemble de mesures qui permettent de comparer ses performances avec d'autres systèmes du même type. Les mesures prises en considération pour estimer la performance sont :
  1. La précision : cela nous permet de comparer les services retournés par le système de découverte et composition qui répondent aux exigences du client, avec ceux non retournés par le système de découverte et composition et qui répondent également aux exigences du client.
  2. Le rappel : Il nous permet de comparer les services retournés par le système de découverte et composition qui répondent aux exigences du client et ceux retourné par le système de découverte et composition qui ne répondent pas aux exigences du client.
  3. F-mesure : représente la moyenne pondérée de la précision et du rappel. F-mesure est compris entre 0 et 1.

4. Temps de réponse : cela permet d'évaluer le temps que prend le système de découverte et composition dès la réception de la requête du client jusqu'à l'envoi des services composites à ce dernier.
5. Temps d'indexation : cela permet d'évaluer le temps que prend le système de découverte pour indexer les nouveaux services recueillis automatiquement ou enregistrés par les fournisseurs des services.

Enfin, on attribue un score à chaque mesure prise en considération par le système de découverte et composition.

- Utilisation d'exemples réalistes du monde réel : le système de découverte et composition qui utilise des bases de services issues du monde réel indique qu'il minimise le risque d'ignorer quelques caractéristiques importantes des services Cloud. Cela indique également que le système de découverte est plus adapté aux technologies de développements de services Cloud, adoptées par les fournisseurs de services, contrairement aux systèmes de découverte qui n'utilisent que des services artificiels ou conçus manuellement pour faire les tests [156].
- Richesse sémantique des descriptions : les services contenus dans la base des tests doivent être décrits en détail afin d'avoir une découverte sémantiquement efficace. Les systèmes de découverte qui sont testés sur des services sémantiquement annotés sont considérés plus crédibles que les systèmes qui ne sont testés que sur des services dépourvus de toute description sémantique [156].
- Indépendance entre descriptions et requêtes : quand un client introduit sa requête auprès d'un système de découverte de services, il la formule indépendamment de la description du service qu'il désire i.e. sans qu'il sache les détails des descriptions des services. Un système de découverte de service qui utilise un ensemble de requêtes variées et indépendantes des descriptions des services, est plus crédible qu'un système qui utilise des requêtes destinés aux descriptions des services contenus dans la base des tests [156].
- Catégorie des services choisis pour les tests : Nous définissons ce critère qualitatif comme la diversité des domaines présentés dans la base de services. On attribue un score à chaque catégorie prise en considération par le système de découverte et composition.
- Méthodologie d'évaluation : Nous définissons ce critère qualitatif comme le font les experts qui ont testé le système de découverte et composition. Les types d'experts pris en considération pour estimer la performance sont :

  1. Groupe de spécialiste neutres.
  2. Développeurs du système de découverte et composition.

  Si le système a été évalué par les deux types d'experts alors il est «hautement crédible», si le système a été évalué seulement par le premier type d'expert alors il est «moyennement crédible», si le système a été évalué seulement par le deuxième type d'experts alors il est «faiblement crédible».

VII.5. Survol et Comparaison

Dans cette section nous rappelons brièvement les travaux vus dans le chapitre 3 «Travaux liés à l'approche proposée». Ensuite, pour chacun nous considérons seulement les critères de comparaison manquants et communs :

VII.5.1. Découverte et composition de services web

Dans l'article [87], les auteurs ont proposé un système d'extraction de mots clés sémantiques basé sur un algorithme de subsomption qui compare les entrées/sorties et la requête. Dans l'article [88], les auteurs ont proposé une approche basée-vecteur d'extraction et WordNet, qui permet de réduire la taille et l'espace de recherche. Dans l'article [89], les auteurs ont proposé une approche de découverte et sélection de services basée sur la méthode d'encodage «Prufer» , afin de présenter les services découverts. Dans l'article [90], les auteurs ont proposé un nouveau modèle de découverte et composition de services Web basé sur les réseaux sociaux (nommé LinkedWS). Dans l'article [91], les auteurs ont proposé un mécanisme basé-IR-Style pour la découverte et composition de services web en introduisant la notion de degré de préférence. Dans l'article [93], les auteurs ont proposé une approche sémantique basée sur les aspects structuraux et sémantiques en utilisant une ontologie basée OWL-S pour la découverte de services web.

Ces approches ne prennent pas en considération :

1. La réduction du temps de réponse en adoptant la notion de clustering qui permet de réduire l'espace de recherche.
2. La combinaison de différents éléments de matching sur chaque étape des algorithmes proposés.

VII.5.2. Découverte et sélection de SaaS

Dans les articles [94], [95] et [96], les auteurs ont proposé des Framework basés-mécanismes hiérarchiques pour la classification de Software-as-a-Service, afin de sélectionner les meilleurs services. Dans les articles [97] et [98], les auteurs ont proposé des méthodes non structurées basées P2P pour la découverte de SaaS. Dans l'article [99], les auteurs ont proposé un Framework basé attribut dynamique et services web pour la représentation de services et ressources dans le Cloud. Dans l'article [100], les auteurs ont proposé un Framework orienté service pour la réalisation d'un Cloud broker, qui sert d'intermédiaire entre client et service Cloud. Dans l'article [101], les auteurs ont proposé un Framework basé ontologie Cloud pour l'enregistrement et la découverte de service Cloud. Dans l'article [102], les auteurs ont proposé un système de publication, découverte, et sélection de Software-as-a-Service basé sur une nouvelle ontologie Cloud qui regroupe 650 concepts. Dans l'article [103], les auteurs ont proposé un comparateur systématique entre fournisseurs de services Cloud nommés CloudCmp. Dans l'article [104], les auteurs ont proposé un modèle de planification de ressources dans les entreprises de production de Software-as-a-Service basé sur la qualité de ces derniers.

Ces approches ne prennent pas en considération

1. La combinaison de différents éléments de matching sur chaque étape des algorithmes proposés.

2. La création de classes de catégorisation pour améliorer la précision des résultats retournés au client.
3. La composition de services ce qui les rend moins efficaces pour le traitement de requêtes complexes.

VII.5.3. Composition de SaaS

Dans l'article [105], les auteurs ont proposé une approche de composition de services sur le Cloud en utilisant IC Cloud (Imperial college Cloud) comme exemple de démonstration. Dans l'article [106], les auteurs ont proposé une approche de composition de services Cloud, représentée sous forme de services web, et basée sur la notion de Skyline (ligne d'horizon). Dans l'article [107], les auteurs ont proposé une approche basée-partage d'informations et modélisation ontologique pour la composition et l'approvisionnement de services Cloud. Dans l'article [108], les auteurs ont proposé une approche de recherche et composition de service Cloud basé sur les aspects fonctionnels et non fonctionnels, en exploitant WordNet comme moyen de calcul de degrés de similarité sémantique .Dans l'article [109], les auteurs ont proposé la notion de service de composition virtuel , ce genre de service sert à lier deux services exigés par le client et qui ne sont pas composables. Dans l'article [110], les auteurs ont proposé un modèle abstrait d'approvisionnement, recherche et composition de services fédérés dans le Cloud, en prenant en compte l'aspect sémantique et QoS de ces services. Dans l'article [111], les auteurs ont proposé un modèle d'exploitation et de composition de services sécurisés dans le Cloud. Dans l'article [112], les auteurs ont proposé une approche de composition de service Cloud basé-attributs QoS. Dans l'article [113], les auteurs ont proposé une approche sémantique de composition de service Cloud présentée sous forme de service web. Dans l'article [114], les auteurs ont proposé une approche d'optimisation des attributs QoS basée sur une représentation arborescente des services Cloud. Dans l'article [115], les auteurs ont proposé un modèle de service sous forme de flux, qui a pour but d'être accessible et exploitable par le client à n'importe quel endroit et à n'importe quel moment.

Ces approches ne prennent pas en considération :

1. La réduction du temps de réponse en adoptant la notion de clustering qui permet de réduire l'espace de recherche.
2. La création de classes de catégorisations pour améliorer la précision des résultats retournés au client.
3. La réduction des tests de connectivité afin de réduire le temps de réponse.

Enfin le tableau VII.1 illustre les principales différences et similarité entre les algorithmes de matching proposés et les algorithmes vus dans le chapitre 4.

VII.5.4. Architecture basé-Cloud pour la découverte et composition de services

Dans l'article [116], les auteurs ont proposé un modèle basé Cloud appelé «Service Registry on the Cloud(SRC)», qui représente une extension du modèle basé mots clés déployé comme une application sur le Cloud. Dans l'article [117], les auteurs ont proposé une approche Cloud computing basée «open standard Extensible Messaging and Présence Protocol (XMPP)» appliquée au domaine de la bioinformatique. Dans l'article [118], les

auteurs ont proposé une architecture basée Cloud qui implémente une ontologie de découverte pour aider les clients à choisir les meilleurs services selon leur QoS. Dans l'article [119], les auteurs ont proposé un schéma qui permet d'optimiser l'évolutivité des services dans le Cloud en utilisant la composition comme moyen d'assurer cela. Dans l'article [120], les auteurs ont proposé une approche de composition de services à la demande sur le Cloud en exploitant leur caractère dynamique.

Ces approches ne prennent pas en considération

1. La gestion de ressources virtuelles et logiques sur la plateforme supportant les architectures proposées.
2. La gestion d'erreurs et changement dans le réseau i.e. absence de système de sauvegarde qui assure la robustesse des architectures proposées.
3. La mise à jour automatique après chaque période de temps.

VII.5.5. Architecture basé-Agent pour la découverte et composition de services

Dans l'article [121], les auteurs ont proposé une approche basée agent associée à un algorithme de matching basé QoS, pour la sélection et le tri de services web. Dans les articles[122] et [123], les auteurs ont proposé des approches Grid basées agent mobile pour la découverte et sélection de services. Dans l'article [124], les auteurs ont proposé une architecture basée agents situés et mobiles pour la découverte et sélection de services web.

Ces approches ne prennent pas en considération :

1. La gestion des ressources virtuelles et logiques de leurs plateformes.
2. La robustesse et la sauvegarde des données et applications du système de découverte.

VII.5.6. Cloud basé-agent pour la découverte et composition de services

Dans l'article [125], les auteurs ont proposé une architecture basée agent pour la composition de services dans le Cloud. Les auteurs ont proposé également l'utilisation de services web comme interface pour les services Cloud. Dans l'article [126], les auteurs ont proposé une architecture Cloud basée-agent mobile et situé pour la découverte et sélection de services web. Dans l'article [127], les auteurs ont proposé un modèle automatique de construction de services composites basés intégration d'agent, Cloud et sémantique. Dans l'article [128], les auteurs ont proposé un moteur de recherche de services dans le Cloud basé description ontologique.

Ces approches ne prennent pas en considération :

1. La gestion des ressources virtuelles et logiques de leurs plateformes.

Enfin le tableau VII.2 illustre les principales différences et similarité entre l'architecture proposée et les architectures vues dans le chapitre 4.

VII.6. Conclusion

Dans ce chapitre nous avons proposé un nouveau modèle de comparaison entre approches de découverte et composition de services SaaS. Le modèle proposé adopte 7 vues différente, 4 classe d'approches et 27 critères de comparaisons.

Le modèle nous a également permis de comparer l'approche proposée et 25 algorithmes de matching ainsi que 11 architectures.

La comparaison révèle l'importance de la contribution présentée dans cette thèse par rapport aux autres approches notamment pour les critères d'exactitude et gestion des ressources Cloud.

Critères / Classes	Approches	Clustring	Degré Matching	Création de classes	Matching Multi-étapes	WSDL	Hybridation	WordNet	Composition /sélection	Tests connectivités	Exactitude	Evolutivité	Performance
AL	[87]	×	✓	✓	✓	×	✓	✓	×	×	6	Moyen	0
	[90]	×	✓	×	✓	✓	×	×	✓	✓	8	Faible	1
	[100]	×	✓	×	✓	×	×	×	×	×	4	NON	1
	[101]	×	×	×	×	×	×	×	×	×	4	NON	0
	[102]	✓	✓	×	✓	×	✓	×	×	×	8	Faible	2
	[107]	×	×	×	✓	×	×	×	✓	✓	2	NON	0
	[110]	×	×	×	✓	×	×	×	✓	×	7	NON	0
ALNL	[113]	×	×	×	✓	✓	✓	×	✓	✓	4	Moyen	2
	[88]	×	✓	×	×	✓	×	✓	×	×	6	Moyen	1
	[91]	×	✓	×	✓	✓	✓	×	✓	×	7	Moyen	4
	[89]	×	✓	×	✓	✓	×	×	×	×	5	Moyen	4
	[93]	×	✓	✓	×	✓	×	×	×	×	3	Faible	2

ANL	[94, 95]	×	✓	×	×	×	×	×	×	×	1	NON	0
	[97, 98]	×	✓	×	✓	✓	×	×	×	×	4	Faible	0
	[99]	✓	✓	×	✓	✓	✓	×	×	×	5	Moyen	1
	[103]	×	✓	×	✓	×	✓	×	×	×	3	Faible	2
	[105]	×	×	×	✓	✓	×	×	✓	×	8	Faible	0
	[106]	×	✓	✓	✓	×	×	×	✓	✓	5	NON	2
	[108]	×	✓	×	✓	✓	✓	✓	✓	×	5	Haute	2
	[109]	×	×	×	✓	✓	×	×	×	✓	2	Faible	0
	[111]	×	×	×	✓	×	✓	×	✓	×	3	Faible	0
	[112]	×	✓	×	✓	×	×	×	✓	✓	4	NON	2
	[114]	×	✓	×	✓	✓	×	×	✓	✓	8	Faible	2
	Approche proposée	✓	✓	✓	✓	✓	×	✓	✓	✓	**15**	**Moyen**	**1**

Tableau VII.1 : Classification des algorithmes de découvertes et compositions de services

Critères / Approches	Recueils automatique	Mis-à-jour automatique	Robustesse	Présence d'Indexe	modele de payement	Gestion de ressources
[116]	×	✓	✓	✓	✓	1
[118]	×	×	×	✓	✓	0
[120]	×	×	×	✓	✓	0
[121]	✓	×	×	✓	×	0
[122, 123]	✓	×	×	×	✓	0
[124]	✓	×	×	×	×	1
[125]	✓	✓	×	✓	✓	2
[126]	✓	×	×	✓	✓	1
[127]	×	×	×	✓	✓	1
[128]	×	×	×	×	✓	0
Approche proposée	✓	✓	✓	✓	✓	**7**

Tableau VII.2 : Classification des architectures de découvertes et compositions de services

VIII
Conclusion et perspectives

Le Cloud computing est une technologie qui permet aux clients ainsi qu'aux fournisseurs de services de bénéficier de capacités de traitement illimitées, avec des coûts d'utilisation et de déploiement très compétitifs. Avec le Cloud computing, le fournisseur de services est le seul responsable du déploiement et gestion de son infrastructure, ainsi que la publication de ces caractéristiques. Ces caractéristiques sont prises en considération par un système intermédiaire qui sert de moyen pour la découverte et composition de services SaaS. Ce système doit gérer ces propres ressources logiques et virtuelles intelligemment et efficacement, afin de pouvoir exécuter les algorithmes de matching responsables de la sélection et trie de services SaaS. Enfin, le client qui représente la cible de tout fournisseur de service, utilise le système de découverte et composition pour choisir les meilleurs services, qui correspondent à ces exigences fonctionnelles, non-fonctionnelles et Cloud.

Afin de pouvoir arriver à un système de découverte et composition de services complet, il est nécessaire de proposer des modèles de matching et de gestion de ressources, qui permettent de couvrir divers aspects. A cette fin, nous avons divisé notre contribution en trois aspects :

(1) Aspect matching : il concerne la proposition d'algorithmes de matching pour calculer le degré de correspondance entre requête et description de services SaaS,

(2) aspect architectural : il concerne la proposition d'un modèle pour le moteur de découverte et composition de services, ainsi qu'un ensemble de mécanismes de gestion de ressources interne et externe,

(3) Aspect comparatif : il concerne la proposition d'un modèle de comparaison entre approches de découverte et composition de services Cloud, pour nous permettre de comparer l'approche proposée avec celles déjà existantes, afin de pouvoir dégager les avantages et les inconvénients du travail proposé, et de pouvoir l'optimiser à l'avenir. Dans le travail présenté sur cette thèse nous comptons 22 contributions élémentaires :

Sur l'aspect matching :

1. Proposition d'un nouveau modèle de description de services SaaS qui prend en compte les attributs de catégorisation, fonctionnels, non-fonctionnels et Cloud des services SaaS.
2. Proposition de la notion du «degré d'existence», qui permet non seulement le calcul du degré de correspondance entre une forme textuelle et une forme arborescente, mais également la préservation des deux formes intactes, ce qui permet de réduire le temps de réponse et préserve la sémantique.
3. Proposition d'un nouvel algorithme de sélection de services SaaS, basé sur la notion du degré d'existence, qui combine les attributs de catégorisations et fonctionnels, mais surtout, qui lie la phase de sélection à celle de composition en proposant quelques classes d'existences, qui servent de moyen de catégorisation de services SaaS afin de les préparer à la phase de composition.
4. Proposition d'un nouvel algorithme de composition et de tri de services SaaS, qui combine les attributs fonctionnels et non-fonctionnels pour la création de services composites.

5. Proposition d'un ensemble de règles de compositions de services, qui servent comme moyen pour réduire le nombre de tests de connectivités en prenant compte la convergence vers des services composites optimaux (avoir les meilleures combinaisons de services SaaS sélectionnés).
6. Proposition d'un nouveau modèle de coût pour le calcul de distance entre arbre WSDL qui se base sur le gain ou la perte de la sémantique de la requête.
7. Utilisation de la requête du client comme guide sur toutes les phases des algorithmes de matching proposés (y compris la phase de composition), afin d'assurer que les résultats retournés au client soient aussi conformes que possible à ces exigences fonctionnelles, non fonctionnelles et Cloud.
8. Enfin, la proposition d'un mécanisme de trie et classification de services SaaS qui combine les attributs Cloud, non-fonctionnels et fonctionnels.

Sur l'aspect architectural :

1. Une nouvelle architecture Cloud basée agent pour la découverte et composition de services SaaS.
2. Un ensemble de modèles basés agents pour la gestion de ressources logiques et virtuelles sur les nœuds Cloud du moteur proposé.
3. Un système basé agent mobile pour assurer la liaison entre les ressources du moteur proposé.
4. La notion de table d'auto-approvisionnement de ressources, qui sert de moyen de description des ressources du moteur proposé.
5. Un nouveau mécanisme assurant une gestion globale des composants Cloud du moteur proposé.
6. Un nouveau mécanisme de gestion d'espace de stockage.
7. Un nouveau mécanisme pour la virtualisation de ressources physiques sur chaque nœud Cloud.
8. Un nouveau mécanisme pour la distribution et l'affectation de machines virtuelles.
9. Un nouveau mécanisme de gestion de ressources externes sur le Cloud.
10. Un nouveau mécanisme de prédiction de besoins en ressources.
11. Enfin, un ensemble de mécanismes de gestion des instances responsables de l'exécution des algorithmes de matching.

Sur l'aspect comparatif :

1. Une nouvelle taxonomie pour les approches de découverte et composition de services SaaS.
2. Un modèle de critères permettant l'évaluation et la comparaison entre les approches de découverte et composition de services SaaS.
3. Présentation de deux tableaux comparatifs, qui résument les principales différences et similarités entre 32 approches de découverte et composition de services.

Dans les travaux futurs nous visons à ajouter l'aspect infrastructure aux algorithmes de matching proposés afin de pouvoir évaluer les capacités de chaque infrastructure Cloud sur

laquelle est déployée le service SaaS demandé par le client. Cela nous permettra d'améliorer l'exactitude des algorithmes proposés et offrir plus de possibilité et de choix aux clients.

Nous comptons également étendre l'architecture proposée en ajoutant un module ou un agent responsable de la conception collaborative automatique de services SaaS [163], et cela à la demande du client (s'il exprime sa non-satisfaction des services qui lui sont retournés par le moteur proposé).

Enfin, nous visons à améliorer le modèle de comparaison proposé, par l'ajout d'un système de classification, destiné spécialement aux approches de matching pour la découverte et composition de services SaaS, car le système adopté sur le chapitre 7 n'est destiné qu'aux approches sémantiques de découverte de services Web.

Annexe A.
Caractéristiques d'infrastructure des services Cloud

Les caractéristiques Cloud réprésentées dans le modèle de déscription de services cloud sur la Section 3 du chapitre IV, ne représente pas les seuls caractéristiques entre services cloud et services web, mais il existe un autre genre d'attributs qui distinguent ces deux derniers. Ces attributs décrivent l'infrastructure Cloud computing sur laquelle les services SaaS sont déployés. Ainsi on peut utiliser les attributs d'infrastructure afin de raffiner les résultats de seléction et composition de services. Ce raffinement permettra alors de choisir les services SaaS qui sont déployés sur les meilleures infrastructure-as-a-Service. Après avoir étudié les travaux [82] [103] [139] [142] et [164] qui décrivent les caractéristiques des Infrastructure-as-a-Service, nous avons recueilli 43 attributs (qui représentent les plus importants et les plus communs). Ensuite, nous les avons classés en trois grandes catégories : statique, dynamique, et pré-déterminée. Nous avons également classé les caractéristiques de chaque catégorie selon la nature de leurs valeurs: valeurs numériques, valeurs textuelles et valeurs pré-déterminées. Les attributs statiques représentent les caractéristiques d'infrastructure que le fournisseur de services change rarement (ou après une longue période) après les déploiements de ses services IaaS sur Internet. Les attributs dynamiques représentent les caractéristiques d'infrastructure qui changent en fonction de la fréquence d'utilisation des ressources IaaS par ses clients. Enfin, les attributs pré-déterminés représentent les caractéristiques d'infrastructure fixés par le fournisseur de service IaaS pour décrire les options et les politiques adoptées pour le déploiement de sont infrastructure. Le tableau A.1 montre les attributs d'infrastructure collectées et leurs catégories.

Catégories	**Types de valeurs**	**Caractéristiques**	**Déscription / valeur**
Statique	Valeur numérique	Nombre total de noyaux	Entier
		Vitesse-noyau (MHZ)	Réel
		Nombre total de CPU	Entier
		Vitesse-CPU (MHZ)	Réel
		Nombre de CPUs virtuel	Entier
		Capacité totale de la RAM (GB)	Réel
		Capacité totale des disques (GB)	Réel
		Temps de démarrage	Réel
		Temps de déploiement	Réel
		Version du système d'éxploitation	Convertie en Entier
		Nombre d'instances	Nombre d'instances qui peuvent être créées sur l'IaaS pour le service cloud (Entier)
		Capacité de la bande passante (Gbps)	Réel
		Latence du réseau	Réel
		Nombre de noeuds	Nombre total de noeuds sur le cluster (Entier).

	Valeur textuelle	Type de la Machine	Les types des machines utilisées pour le déploiement des infrastructures IaaS (e.x. IBM Flex System p260 Compute Node).
		Système d'éxploitation	Le nom du système d'éxploitation utilisé pour déployer les machines virtuelles.
		Technologie de virtualisation	Le nom du système utilisé pour créer les machines virtuelles.
		Nom de l'hébergeur	Le nom de la companie qui offre l'infrastructure IaaS qui héberge le service Cloud.
		Système de fichier	Le système utilisé pour stocker les fichiers sur les disques durs de l'IaaS e.x. NTFS, FAT, ext4 … etc.
		Les applications et les framworks pris en charge	Les applications et frameworks offerts aux clients pour configurer leurs MVs.
		Outils de récupération en cas de catastrophe	Les noms des outils utilisés pour faire face aux catastrophes.
		Outil de contrôle et rapport dynamique	Les noms des outils utilisés pour superviser les ressources de l'IaaS.
		Outils de test des performances	Les noms des outils utilisés pour tester les performances des ressources IaaS.
		Outils d'équilibrage de charge	Les noms des outils utilisés pour distribuer équitablement la charge du travail sur les MVs.
		outils d'auto-mise à l'échelle	Les noms des outils utilisés pour augmenter ou réduire la tail des resources de l'IaaS.
		Dispositifs de refroidissement	Les noms des dispositifs utilisés pour refroidir les Datacenters de l'IaaS.
		Equipements réseau pour l'accès Internet	Les noms des dispositifs utilisés pour lier l'IaaS à Internet.
		Type du CDN (Content Delivery Network)	Le type du CDN utilisé pour reduire les coûts de la bande passante et rendre disponible tout type de contenu ou d'information aux clients (e.x. CDN de diffusion multimédia).
Dynamique	Valeur numérique	CPU en cours d'exécution	Entier
		Pourcentage d'utilisation des CPUs	Pourcentage de CPU utilisé dans tout le cluster
		Espace libre de la RAM (GB)	Réel
		Pourcentage d'espace libre dans la RAM	Pourcentage de mémoire RAM libre sur tout le cluster

		Espace libre des disques (GB)	Réel
		Pourcentage d'espace libre des Disques	Pourcentage d'espace disque libre sur tout le cluster
Pré-déterminée	valeur pré-déterminée	Architecture-Hardware	(i) 32-bit (ii) 64-bit
		Politique du système d'éxploitation	(i) Single OS support (ii) Multiple OS support
		Politique de suppression des données	(i) Deleted (ii) Made inaccessible
		Support pour le backup (préservation des données)	(i) Yes ((i) Daily (ii) weekly) (ii) No
		Support pour une politique de cryptage de données	(i) Yes ((i) Encryption algorithm used (ii) Key strength) (ii) No
		Support pour une politique de conformité	(i) Yes (ii) No
		Support pour la séparation de machine virtuelle	(i) Yes (ii) No
		Support pour l'authentication des clients	(i) Yes (ii) No
		Support pour système de refroidissement	(i) Yes (ii) No

Tableau A.1 : Résumé des caractéristiques d'infrastructure des services SaaS

Annexe B
Liste des publications

B.1. Revues Internationales

1. Saouli, H., Kazar, O. and Benharkat, A.N. (2012) 'A Cloud Computing Framework Based Mobile Agents for Web Services Discovery and Selection', *Int. J. of Emerging Trends & Technology in Computer Science (IJETTCS)*, Vol. 1, No. 02, pp. 171-189.
2. Saouli, H., Kazar, O. and Benharkat, A.N. (xxxx) 'SaaS-DCS: software-as-a-service discovery and composition system-based existence degree', *Int. J. Communication Networks and Distributed Systems*, Vol. X, No. Y, pp.xxx–xxx. ***(Accepté et en ligne)***
3. Saouli, H., Kazar, O. and Benharkat, A.N. (xxxx) 'SaaS-SRS: Software-as-a-Service (SaaS) description, selection and ranking system-based documentation tag', *frontier of computer science in china*, Vol. X, No. Y, pp.xxx–xxx. ***(En cours de review " Major revision")***
4. Saouli, H., Kazar, O., Benharkat, A.N. and Bourekache S. (xxxx) 'ASDCE-CS: AGENT-BASED SERVICE DISCOVERY AND COMPOSITION ENGINE IN CLOUD COMPUTING SYSTEM', *J. of Web Engineering*, Vol. X, No. Y, pp.xxx–xxx. ***(En cours de review)***

B.2. Conférences Internationales

1. Saouli, H., Kazar, O., Benharkat, A.N. Amghar, Y. (2012) 'A Cloud computing approach based on mobile agents for web services discovery', *Proceedings of the Second International Conference on Innovative Computing Technology (INTECH 2012)*, Casablanca, Morocco, pp. 297-304.
2. Saouli, H., Kazar, O. and Benharkat, A.N. (2012) 'A New Cloud computing framework based on mobile agents for web services discovery and selection', *Proceedings of the 13th International Arab Conference on Information Technology (ACIT'2012)*, Zarqa University, Jordan, pp. 587-594.
3. Benfenatki, H., Saouli, H., Benharkat, A.N. Ghodous, P. Kazar, O. and Amghar, Y. (2013) 'Cloud Automatic Software Development', *Proceedings of the 20th International Conference on Concurrent Engineering*, Melbourne, Australia, pp. 40-49.
4. Saouli, H., Kazar, O. Benharkat, A.N. and Merizig A. (xxxx) 'Cloud Service Selection and Ranking Algorithms based Software and Infrastructural Characteristics', *Proceedings of the 1ère Conférence sur l'Ingénierie Informatique (C2i)*, Algiers, Algeria, pp.xxx–xxx. ***(Accepté en attente de présentation)***

B.3. Workshops Internationaux

1. Saouli, H., Kazar, O. Kankaanpää, J.T. Benharkat, A.N. and Benfenatki, H. (2013) 'An Intelligent Cloud Computing System Based Agent for web service discovery and composition', *Proceedings of the First workshop on Artificial Intelligence and ICT Information Communication Technologies*, Biskra, Algeria.

B.4. Conférences Nationales

1. Saouli, H., Kazar, O. and Benharkat, A.N. (2012) 'An approach Cloud computing based mobile agents for discovery of web services, *Proceedings of the 2nd Artificial Intelligence Doctorial*', Algiers, Algeria.
2. Saouli, H., and Kazar, O. (2012) 'Multi-Agent system Based Web Services Discovery and Selection in Cloud Computing Environment', *Proceedings of the 2nd Doctorial days on informatics*, Guelma, Algeria.

Bibliographie

[1] Brustoloni, J. C. (1991) 'Autonomous agents: characterisation and requirements', *Carnegie Mellon Technical Report CMU-CS-91-204*, Carnegie Mellon University, Pittsburgh.

[2] Smith, D. C., Cypher, A. and Spohrer, J. (1994) 'Kidsim: programming agents without a programming language', *Communications of the ACM*, Vol. 37, No. 7, pp. 54–67.

[3] Maes, P. (1994) 'Agents that reduce work and information overload', *Communications of the ACM*, Vol. 37, No. 7, pp. 30-40

[4] Hayes-Roth, B. (1987) 'Blackboard systems', *Encyclopedia of Artificial Intelligence*, John Wiley Sons, ISBN-13: 978-0471807483

[5] http://www.networking.ibm.com/wbi/wbisoft.htm

[6] Mlungisi, d. (2008) 'Agents, Agent architectures and Multi-agent systems', Master of science, Ehlers, E.M., Oosthuizen, O.L., Johnnesburg, 143 p.

[7] Sycara, K.P. (1998) 'The many faces of agents', *Al Magazine*, Vol. 19, No. 2, pp. 11-12.

[8] Sycara K.P. (1998) 'Miltu-agent system', *Al Magazine*, Vol. 19, No. 2, pp. 79–92.

[9] Gray, R. S., Kotz, D., Nog, S.,Rus, D., and Cybenko, G. (1996) 'Mobile Agents for Mobile Computing', *Technical Report PCS-TR96-285,* Department of Computer Science, Dartmouth College.

[10] Nguyen, A., Stewart, I. and Yang, X. (1998) 'A Mobile Agent Model: Applications for E-Commerce' *Proceedings of Seventh Australian World Wide Web Conference*, Brisbane Australia, pp. 21 - 25.

[11] Jeffrey J.P.T. and Lu. M (2006) 'Security Modeling and Analysis of Mobile Agent Systems', *SERIES IN ELECTRICAL AND COMPUTER ENGINEERING*, World Scientific Publishing Company, ISBN-13: 978-1860946349.

[12] Baumann, J. (2000) 'Mobile Agents: Control Algorithms', *Lecture Notes in Computer Science,* Springer Berlin Heidelberg, ISBN: 978-3-540-41192-5.

[13] http://www.gartner.com/technology/home.jsp

[14] Nagappan, R., Skoczylas R., Sriganesh, R. P. (2003) 'Developing Java™ Web Services: Architecting and Developing Secure Web Services Using Java', Wiley Publishing Inc, ISBN-13: 072-3812236404.

[15] Newcomer, E. (2000) 'Understanding Web Services- XML, WSDL, SOAP and UDDI', Addison-Wesley Professional, ISBN-13: 078-5342750812.

[16] Mather, T. Kumaraswamy, S. and Latif, S. (2009) 'Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance', O'Reilly Media, ISBN-13: 978-0596802769.

[17] http://fr.wikipedia.org/wiki/Fournisseur_de_services_d%27applications

[18] http://www.microsoft.com/china/ard/en/innoforum/innoforum_11.mspx

[19] http://www.boingboing.net/2009/09/02/cloud-computing-skep.html

[20] Shuai, Z. and Shufen, Z. (2010) 'Cloud computing research and development trends', *Proceedings of 2010 Second International Conference on Future Networks*, Sanya, Hainan, China, pp. 93-97

[21] http://www.vmware.com/cloud-computing/overview

[22] http://www.emc.com/digital_universe

[23] Eric, A. and Lozano, M.B. (2010) 'Executive's Guide to Cloud Computing', John Wiley & Sons, ISBN-13: 978-0470521724

[24] Vaquero, L. M., Rodero-Merino, L., Caceres, Juan. and Lindner, M. (2009) 'A Break in the Clouds: Towards a Cloud Definition', *ACM SIGCOMM Computer Communication Review*, Vol. 39, No. 1, pp. 50-55.

[25] Mell, P. (2011) 'The NIST Definition of Cloud Computing' *Timothy Grance Special Publication 800-145*, National Institute of Standards and Technology, Gaithersburg.

[26] http://blogs.idc.com/ie/?p=190

[27] www.451group.com/reports/executive_summary.php?id=619

[28] Yuanshun, D, Yanping, X. and Gewei, Z (2009) 'Self-healing and Hybrid Diagnosis in Cloud Computing', *Proceedings CloudCom of 1st International Conference on Cloud Computing*, Beijing, China, pp. 45-56.

[29] http://whatiscloud.com/cloud_characteristics/multi_tenancy

[30] http://www.expertglossary.com/cloud-computing/definition/linearly-scalable

[31] Kung-Kiu, L., Winfried, L. and Ernesto, P. (2013) 'Service-Oriented and Cloud Computing', *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, ISBN: 978-3-642-40651-5.

[32] Buyya, R., Garg, S.K. and Calheiros, R.N. (2011) 'SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions', *Proceedings of the International Conference on Cloud and Service Computing (CSC)*, Hong Kong, China, pp. 1-10.

[33] www.broadcom.com/collateral/wp/Virtualization-WP100-R.pdf

[34] http://www.hosting.com/resources/white-papers/what-cloud-computing-means-to-you-efficiency-flexibility/success-cloud-computing-means/

[35] http://www.bibsonomy.org/bibtex/2a5228c1a4057026333619d53f753e449/vonposer

[36] https://cloud.google.com/developers/articles/building-high-availability-applications-on-google-compute-engine/

[37] http://en.wikipedia.org/wiki/Platform_as_a_service

[38] http://en.wikipedia.org/wiki/EMC_Corporation

[39] Hofmann, P. and Woods, D. (2010) 'Cloud Computing: The Limits of Public Clouds for Business Applications' *IEEE Internet Computing*, Vol. 14, No. 6, pp. 90–93.

[40] Milojicic, D. and Wolski, R. (2011) 'Eucalyptus: Delivering a Private Cloud. *Computer*, Vol. 44, No. 4, pp. 102–104.

[41] http://www.strategicitarchitecture.com/2009/11/flight-plan-deploying-the-cloud/

[42] Mathur, P. (2010) **'**Cloud Computing: New challenge to the entire computer industry', *Proceediongs of the 1st International Conference on Parallel, Distributed and Grid Computing*, Solan, India, pp. 223-228.

[43] Jian J.Z., Chengdu, C. and Nan, Zhang. (2011) 'Cloud Computing-based Data Storage and Disaster Recovery', *Proceediongs of the* International Conference on Future computer science and education, Xi'an, China, pp. 629-632.

[44] Hurwitz, J., Robin, B., Kaufman, Marcia. and Halper, F. (2010) 'Cloud Computing for Dumies', Wiley Publishing, ISBN-13: 978-0470484708.

[45] https://appengine.google.com/

[46] http://azure.microsoft.com/fr-fr/

[47] https://aws.amazon.com/fr/ec2/?nc1=h_ls

[48] www.ibm.com/cloud-computing/fr/fr/

[49] Caryer, G., Rings, T., Gallop, J. and Schulz, S. (2009) 'Grid/cloud computing interoperability, standardization and the Next Generation Network (NGN)', *Proceeding of 13th International Conference on Intelligence in Next Generation Networks,* Bordeaux, France, pp. 1-6.

[50] http://public.deloitte.no/dokumenter/2_Cloud_Computing[1].pdf

[51] http://www.marketsandmarkets.com/Market-Reports/cloud-computing-234.html, 2010.

[52] http://www.reportlinker.com/p0293136/Cloud-Computing-SaaS-PaaS-IaaS-Market-Mobile-Cloud-Computing-M-A-Investments-and-Future-Forecast-Worldwide.html

[53] http://www.techsciresearch.com/global-cloud-computing-servicemarket-outlook-2014.

[54] http://wintergreenresearch.com/reports/cloudSaaS.html

[55] https://www.gartner.com/doc/411075/gartner--it-spending-staffing

[56] Armbrust, M., Fox, Armando., Griffith, R., Anthony, D.J., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I. and Zaharia, M. (2009) 'Above the Clouds: A Berkeley View of Cloud Computing'. *Report, Electrical Engineering and Computer Sciences,* University of California at Berkeley, USA.

[57] Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I. and Zaharia, M. (2010) 'A view of cloudcomputing' *Communications of the ACM,* Vol. 53 No. 4, pp. 50–58.

[58] http://www.trendmicro.com.cn/cloud-content/us/pdfs/about/2012_global_cloud_security_survey_executive_summary.pdf

[59] Badger, Lee., Bohn, R., Chu, S., Hogan, M., Liu, F., Kaufmann, V., Mao, J., Messina, J., Mills, K., Sokol, A., Tong, J., Whiteside, F. and Leaf, D. (2011) 'US Government Cloud Computing Technology Roadmap Volume II Release 1.0.' *Technical report,* National Institute of Standards and Technology (NIST), USA.

[60] http://aws.amazon.com/cloudformation/.

[61] Rochwerger, B., Breitgand, D. and Epstein, A. (2011) 'Reservoir-when one cloud is not enough', *Computer*, Vol. 44, No 3, pp. 1–7.

[62] Metsch, T., and Edmonds, A. (2011) 'Open Cloud Computing Interface – REST ful HTTP Rendering', *Open Grid Forum - OCCI Working group technical report*, Canada.

[63] Metsch, T. Edmonds, A. (2011) 'Open Cloud Computing Interface – Infrastructure' *Open Grid Forum - OCCI Working group technical report*, Canada.

[64] Nyren, R., Edmonds, A., Papaspyrou, A. and Metsch, T. (2011) 'Open Cloud Computing Interface – Core' *Open Grid Forum - OCCI Working group technical report,* Canada.

[65] Crosby, S., Doyle, R., Gering, M. and Gionfriddo, M. (2010) 'Open Virtualization Format Specification', Technical report *Distributed Management Task Force (DMTF).*

[66] http://docs.oasis-open.org/tosca/TOSCA/v1.0/csd04/TOSCA-v1.0-csd04.html

[67] Davis, D. and Pilz, G. (2012) 'Cloud Infrastructure Management Interface (CIMI) Model and RESTful HTTP-based Protocol' *Technical report, Distributed Management Work Force (DMTF).*

[68] http://www.rfc-base.org/rfc-6208.html

[69] http://www.snia.org/cdmi

[70] http://online.wsj.com/article/SB10001424052970204880404577225380456599176.html

[71] http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/

[72] Luo, J., Montrose, B. and Kang, M. (2005) 'An Approach for Semantic Query Processing with UDDI', *Proceedings of Agents, Web Services and Ontologies Merging,* Agia Napa, Cyprus, pp. 1-10.

[73] Singh, M., and Huhns, M. (2005) 'Service-Oriented Computing: Semantics, Processes, Agents', Wiley, ISBN: 978-0-470-09148-7.

[74] http://www.w3.org/TR/wsdl.

[75] Schollmeier, R. (2001) 'A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications', *Proceedings of the First International Conference on Peer-to-Peer Computing,* Linkoping, Suede, pp. 101-102.

[76] Emekci, F., Sahin, O.D., Agrawal, D., and El Abbadi, A. (2004) 'A peer-to-peer framework for Web service discovery with ranking', *Proceedings of the IEEE International Conference on Web Services,* California , USA, pp. 192-199.

[77] Sivashanmugam, K. , Verma, K., and Sheth, A. (2004) 'Discovery of Web services in a federated registry environment', *Proceedings of the IEEE International Conference on Web Services*, California , USA, pp. 270-278.

[78] http://fr.wikipedia.org/wiki.

[79] http://wordnet.princeton.edu/

[80] Bernstein, A. and Klein, M. (2002) 'Discovering Services: Towards High Precision Service Retrieval'. *Proceedings of the CaiSE workshop on Web Services, e-Business, and the Semantic Web: Foundations, Models, Architecture, Engineering and Applications,* Toronto, Canada, pp 260-275.

[81] Menascé, D.A. (2002) 'QoS Issues in Web Services', *IEEE Internet Computing*, Vol. 6, No. 6, pp. 72-75.

[82] Wright, P., Sun, Y.L., Harmer, T., Keenan, A., Stewart, A. and Perrott, R. (2012) 'A constraints-based resource discovery model for multi-provider cloud environments', *Journal of cloud computing*, Vol. 1 No. 6, pp. 1-14.

[83] Fensel, D., Bussler, C. and Maedche, A. (2002) ‘Semantic web enabled web services’, *Sigmod Record ACM*, Vol. 31, No. 4, pp. 1-2.

[84] Gardarin, G. (2002) ‘XML des bases de données aux Services Web’, Dunod edition, ISBN-13: 978-2100069330.

[85] Chakraborty, D. and Joshi, A. (2001) ‘Dynamic service composition: State of the art and research directions’. *Technical report, CS*, Department of Computer Science and Electrical Engineering, University of Maryland, USA.

[86] Medjahed, B. (2004) ‘Semantic Web Enabled Composition of Web Services’, PhD thesis, Barkhi, R., Virginia, USA, 278 p.

[87] Espinoza, M. and Mena, E. (2007) ‘Discovering Web Services Using Semantic Keywords’, *Proceedings of the 5th IEEE International Conference on Industrial Informatics*, Vienna, Austria, pp. 725 - 730.

[88] Chen, L., Yang, G., Wang, D. and Zhang, Y. (2010) 'WordNet powered Web Services Discovery Using Kernel-based Similarity Matching Mechanism', *Proceeding of the Fifth IEEE International Symposium on Service Oriented System Engineering*, Nanjing, China, pp. 64-68.

[89] Algergawy, A., Nayak, R., Siegmund, N., Oppen, V. and Saake, G. (2010) ‘Combining Schema and Level-Based Matching for Web Service Discovery’, *Proceedings of the 10th International Conference, Vienna Austria*, pp. 114–128.

[90] Zakaria, M., Leandro, K.W., Youakim, B., Elnaffar, K.B. and Noura, F. (2011) ‘LinkedWS: A novel Web services discovery model based on the Metaphor of social networks', *Simulation Modelling Practice and Theory*, Vol. 19, No. 1, pp. 121–132.

[91] Yanan, H., Yanchun, Z. and Jinli, C. (2010) ‘Web services discovery and rank: An information retrieval approach', *Future Generation Computer Systems*, Vol. 26, No. 8, pp.1053-1062.

[92] Brin, S. and Page, L. (1998) 'The anatomy of a large-scale hypertextual web search engine', *Computer Networks and ISDN Systems*, Vol. 30, No. 1-7, pp. 107-117.

[93] Amorim, R., Claro, D.B., Lopes, D., Albersand, P. and Andrade, A. (2011) ‘Improving Web service discovery by a functional and structural approach’, *Proceedings of the IEEE International Conference on Web Services*, Washington, DC, USA, pp. 411-418.

[94] Yiming, C. and Yiwei, Z. (2011) ‘SaaS Vendor Selection Basing on Analytic Hierarchy Process’, *Proceedings of the Fourth International Joint Conference on Computational Sciences and Optimization*, Yunnan, China, pp. 511-515.

[95] Godse, M. and Mulik, S. (2009) ‘An Approach for Selecting Software-as-a-Service (SaaS) Product’, *Proceedings of the IEEE International Conference on Cloud Computing*, Bangalore, India, pp. 155-158.

[96] Garg, S.K., Versteeg, S. and Buyya, R. (2013) ‘A Framework for ranking of cloud computing services’, *Future Generation Computer system*, Vol. 29, No. 2, pp. 1012–1023.

[97] Zhou, J., Abdullah, N. and Shi, Z. (2011) ‘A hybrid P2P approach to service discovery in the Cloud’ *International Journal of Information Technology and Computer Science*, Vol. 3, No. 1, pp. 1-9.

[98] Lin, W., Dou, W., Xu, Z. and Chen, J. (2013) 'A QoS-aware service discovery method for elastic cloud computing in an unstructured peer-to-peer network," *Concurrency & Computation: Practice & Experience*, Vol. 25, No. 13, pp. 1843–1860.

[99] Goscinski, A. and Brock, M. (2010) 'Toward dynamic and attribute based publication, discovery and selection for cloud computing', *Future Generation Computer Systems*, Vol. 26, No. 7, pp. 947-970.

[100] Ngan, L.D. (2012) 'OWL-S based semantic cloud service broker', *Proceedings of the 19th International Conference on Web Services (ICWS)*, Honolulu, HI, pp. 560–567.

[101] Tahamtan, A., Beheshti, S.A., Anjomshoaa, A. and Tjoa, A.M. (2012) 'A Cloud repository and discovery framework based on a unified business and Cloud service ontology' *Proceedings of the Eighth World Congress On Services, IEEE*, Honolulu, HI, pp. 203-210.

[102] Afify, Y.M., Moawad, I.F., Badr, N.L. and Tolba, M.F. (2013) 'A Semantic-based Software-as-a-Service (SaaS) Discovery and Selection System', *Proceedings of the 8th International Conference Computer Engineering & Systems (ICCES)*, Cairo, Egypt, pp. 57-63.

[103] Yang, A.L.X., Kandula, S. and Zhang, M. (2010) 'CloudCmp: comparing public cloud providers', *Proceedings of the 10th annual conference on Internet measurement*, New York, USA, pp. 1-14.

[104] Park, J. and Jeong, H-Y. (2012) 'The QoS-based MCDM system for SaaS ERP applications with social network', *Journal of Supercomputing*, Vol. 66, No. 2, pp. 614-632.

[105] Guo, Y-K. and Guo, L. (2011) 'IC Cloud: Enabling Compositional Cloud', *International Journal of Automation and Computing*, Vol. 8, No. 3, pp. 269-279.

[106] Wang, Shangguang., Sun, Q., Zou, H. and Yang, F. (2013) 'Particle Swarm Optimization with Skyline Operator for Fast Cloud-based Web Service Composition', *Mobile Networks and Applications, Vol*. 18, No. 1, pp. 116–121,.

[107] Serrano, M., Shi, Lei., Foghlú, M.Ó. and Donnelly, W. (2013) 'Cloud Services Composition Support by Using Semantic Annotation and Linked Data', *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, Springer Berlin Heidelberg, ISBN: 978-642-37185-1.

[108] Cheng Zeng, Xiao Guo, WeijieOu, and Dong Han, 'Cloud Computing Service Composition and Search Based on Semantic', CloudCom, , pp. 290–300, 2009.

[109] Fu, J., Tu, W.H.M., Baldwin, B.M.J. and Bastani, F.B. (2010) 'Virtual Services in Cloud Computing', *Proceedings of the IEEE 6th World Congress on Services*, Miami, USA, pp. 467-472.

[110] Ma, H., Schewe, K-D. and Wang, Q. (2009) 'An Abstract Model for Service Provision, Search and Composition' *Proceedings of the IEEE Asia-Pacific Services Computing Conference (IEEE APSCC)*, Singapore, pp. 95-102.

[111] Menzel, M., Warschofsky, R., Thomas, I., Willems, C. and Meinel, C. (2010) 'The Service Security Lab: A Model-Driven Platform to Compose and Explore Service Security in the Cloud', *Proceedings of the 6th IEEE World Congress on Services*, Miami, USA, pp. 115-122.

[112] Lianyong, Q., Wanchun, D., Xuyun, Z. and Jinjun, C.(2012) 'A QoS-aware composition method supporting cross-platform service invocation in cloud environment', *Journal of Computer and System Sciences,*Vol. 78, No. 5, pp. 1316–1329.

[113] Xiangbing, Z. and Fang, M. (2012) 'A Semantics Web Service Composition Approach based on Cloud Computing' *Proceedings of the Fourth International Conference on Computational and Information Sciences*, Chongqing, China, pp. 807-810.

[114] Wu, C-S. and Khoury, I. (2012) 'Tree-based search algorithm for web service Composition in SaaS', *Proceedings of the Ninth International Conference on Information Technology- New Generations*, Las Vegas, USA, pp. 132-138.

[115] Zhu, Y. Shtykh, R.Y. and Jin, Q . (2010) 'Provision of Flowable Services in Cloud Computing Environments' *Proceedings of the5th International Conference on Future Information Technology (FutureTech),* Busan, South Korea, pp. 1-6.

[116] Chen H, and Li S. (2010) 'SRC A Service Registry on Cloud Providing Behavior-aware and QoS-aware Service Discovery', *Proceedings of the International Conference on Service- Oriented Computing and Applications (SOCA),* Perth, West Australia, pp. 1-4.

[117] Wagener, J., Spjuth, O., Willighagen, L.E. and Wikberg, J.S. (2009) 'XMPP for cloud computing in bioinformatics supporting discovery and invocation of asynchronous web services', *BMC Bioinformatics*, Vol. 10, No. 279, pp. 1-12.

[118] Dastjerdi, A.V., Gholam, Sayed., Tabatabaei, H. and Buyya, R. (2010) 'An Effective Architecture for Automated Appliance Management System Applying Ontology-Based Cloud Discovery', *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, Melbourne, Australia, pp. 104-112.

[119] Wu, J., Liang, Q. and Bertino, E. (2009) 'Improving Scalability of Software Cloud for Composite Web Services', *Proceedings of the IEEE International Conference on Cloud Computing*, Bangalore, India, pp. 143 - 146.

[120] Zhou, J., Athukorala, K., Gilman, E., Riekki, J. and Ylianttila, M. (2012) 'Cloud Architecture for Dynamic Service Composition', *International Journal of Grid and High Performance Computing*, Vol. 4, No. 2, pp. 1-15.

[121] Rajendran, T. and Balasubramanie, P. (2010) 'An Optimal Agent-Based Architecture for Dynamic Web Service Discovery with QoS' *Proceedings of the 2nd International conference on Computing, Communication and Networking Technologies*, Karur, India, pp. 1-7.

[122] He, Y., Wen, W., Jin, H. and Liu, H. (2005) 'Agent-based Mobile Service Discovery in Grid Computing', *Proceedings of the Fifth International Conference on Computer and Information Technology (CIT'05),* Shanghai, China, pp. 351 - 355.

[123] Aversa, R., Di-Martino, B. and Mazzocca, N. (2004) 'Terminal aware Grid resource and service discovery and access based on Mobile Agents technology', *Proceedings 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing*, Coruña, Spain, pp.40-45.

[124] Jingliang, C. and Zhe, M. (2010) 'Research on Web Service Discovery Based on Mobile Agents' *Proceedings International Conference On Computer Design And Applications*, Qinhuangdao, China, pp. 385-388.

[125] Gutierrez-Garcia, J.O. and Sim, K.M. (2013) 'Agent-based Cloud service composition', *Applied Intelligence*, Vol. 38, No. 3, pp. 436-464.

[126] Saouli H., Kazar O., Benharkat A.N. and Amghar, Y. (2012) 'A Cloud computing approach based on mobile agents for Web services discovery', *Proceedings of the 2nd International Conference on Innovative Computing Technology (INTECH)*, Casablanca, Morocco, pp. 297-304.

[127] José A., Coria, G. and Corachado, J.M. (2014) 'Castellanos-Garzón, Juan M. Corchado, 'Intelligent business processes composition based on multi-agent systems', *Expert Systems with Applications*, Vol. 41, No. 4, pp. 1189–1205.

[128] Kang, J.and Sim, K.M. (2011) 'Cloudle : An Agent-based Cloud Search Engine that Consults a Cloud Ontology', *Proceedings of the International Symposium WISS, and International Workshops CISE, MBC*, Hong Kong, China, pp. 416-427.

[129] Zehua, Z. and Xuejie, Z. (2009) 'Realization of Open Cloud Computing Federation Based on Mobile Agent', *Proceedings of the International Conference on Intelligent Computing and Intelligent Systems*, Shanghai, China, pp. 642-646.

[130] Gaoyun, C., Jun, L., Jian, H. and Zexu, W. (2010) 'SaaAS - The Mobile Agent based Service for Cloud Computing in Internet Environment', *Proceedings of the 6th International Conference on Natural Computation*, Yantai Shandong, china, pp. 2935-2939.

[131] Aversa, R., Martino, B.D. Rak, M. and Venticinque, S. (2010) 'Cloud Agency: A Mobile Agent Based Cloud System', *Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems*, Krakow, Poland, pp. 132 - 137.

[132] Jung, G. and Sim, K.M. (2011) 'Agent-based Adaptive Resource Allocation on the Cloud Computing Environment', *Proceedings of the International Conference on Parallel Processing Workshops*, Taipei, China, pp. 345-351.

[133] Yazır, Y.O., Matthews, C. and Farahbod, R. (2010) 'Dynamic Resource Allocation in Computing Clouds using Distributed Multiple Criteria Decision Analysis', *Proceedings of the 3rd International Conference on Cloud Computing*, Miami, UAS, pp. 91 - 98.

[134] Meera, A. and Swamynathan, S. (2013) 'Agent based Resource Monitoring system in IaaS Cloud Environment', *Procedia Technology*, Vol. 10, pp. 200 – 207.

[135] Clark, K., Warnier, M. and Brazier, F.M.T. (2012) 'An Intelligent Cloud Resource Allocation Service - Agent-Based Automated Cloud Resource Allocation using Micro-agreement', *Proceedings of the 2nd International Conference on Cloud Computing and Services Science*, Porto, Portgal, pp. 37-45.

[136] Gutierrez-Garcia, J. O. and Sim, K.M. (2012) 'GA-based cloud resource estimation for agent-based execution of bag-of-tasks applications', *Information System Frontier*, Vol. 14, No. 4, pp 925-951.

[137] Androcec, D., Vrcek, N. and Seva, J. (2012) 'Cloud computing ontologies: a systematic review', *Proceedings of the 3rd International Conference on Models and Ontology based Design of Protocols, Architectures and Services*', Chamonix/Mont Blanc, France, pp. 9-14.

[138] Zhang, M., Ranjan, R., Haller, A. and Georgakopoulos, D. (2012) 'Investigating decision support techniques for automating cloud service selection', *Proceedings of the 4th International Conference on Cloud Computing Technology and Science (CloudCom*), Taipei, Taiwan, pp. 759- 764.

[139] Höfer, C.N. and Karagiannis, G. (2011) 'Cloud computing services: taxonomy and comparison', *Journal of Internet Service Applications*, Vol. 2, No. 2, pp. 81-94.

[140] Lorena, E., Daniela, B.C., Denivaldo, L. and Patrick A. (2012) 'Discovering cloud services by preconditions and effects for compositions', *Proceedings of the 14th International Conference on Enterprise Information Systems*, Wroclaw, Poland, pp. 264-270.

[141] Fortis, T-F., Munteanu, V.I. and Negru, V. (2012) 'Towards an Ontology for Cloud Services', *Proceeding of the Sixth International Conference on Complex, Intelligent, and Software Intensive Systems*, Palermo, Italy, pp. 787-792.

[142] Joshi, K.P., Yesha, Y. and Finin T. (2014) 'Automating Cloud Services Life Cycle through Semantic Technologies' *IEEE Transaction on Services Computing*, Vol. 7, No. 1, pp. 109-122.

[143] Salton, G. (1986) '*Introduction to modern Information retrieval'*, McGraw-Hill Companies, ISBN:0070544840

[144] Joël, P., Nada, L. and Dunja, M. (2004) 'A Rule based Approach to Word Lemmatization', *Proceeding of the SiKDD multiconference*, Ljubljana, Slovenia, pp. 12-15.

[145] Yong, Z. and Jian-lin, L. (2009) 'Research and Improvement of Search Engine Based on Lucene', *Proceeding of the International Conference on Intelligent Human-Machine Systems and Cybernetics*, Hangzhou, Zhejiang, China, pp.270-273 .

[146] Li, Y., Bandar, Z.A. and McLean, D. (2003) 'An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources', *IEEE Transaction On Knowledge and Data Engineering*, Vol. 15, No. 4, pp. 871–882.

[147] Blanco, E., Cardinale, Y. and Vidal, M-E. (2012) 'Experiences of samplingbased approaches for estimating QoS parameters in the Web Service composition problem', *Int. J. Web and Grid Services*, Vol. 8, No. 1, pp.1–30.

[148] Alexander, L., Markus, K., Jens, N., Stefan, T. and Thomas, S. (2009) 'What's Inside the Cloud? An Architectural Map of the Cloud Landscape'. *Proceedings of the International Workshop on Software Engineering Challenges of Cloud Computing*, Vancouver, Canada, pp. 24-31.

[149] Rodrigo, N.C., Rajiv, R., Anton, B., César, D.R., Rajkumar, B., Calheiros, N.R., Ranjan, R., Beloglazov, A., De-Rose, A.C.F. and Buyya, R. (2011) 'CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms', *Software: Practice and Experience (SPE)*, Vol. 41, No. 1. pp. 23-50.

[150] Saurabh, K.G. and Rajkumar, B. (2011) 'NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations', *Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing*, Victoria, Canada, pp. 105-113.

[151] Wickremasinghe, B., Calheiros, N.R., and Buyya, R. (2010) 'CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications', *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, Perth, Australia, pp. 446-452.

[152] Bellifemine, F., Poggi, A. and Rimassa, G. (1999) 'JADE—a FIPA compliant agent framework', *Proceedings of the 4th international conference and exhibition on the practical application of intelligent agents and multi-agents*, UK, pp. 97–108.

[153] http://www.andreas-hess.info/projects/annotator/

[154] Wang, Y. and Stroulia, E. (2003) 'Flexible interface matching for web-service discovery', *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE)*, Rome, Italy, pp. 147 -156.

[155] Mohebbi, K., Ibrahim, S., Khezrian, M., Munusamy, K. Gholam, S., and Tabatabaei, H. (2010) 'A Comparative Evaluation of Semantic Web service Discovery Approaches', *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services*, Paris, France, pp. 33-39.

[156] KÄuster, U., Lausen, H. and KÄonig-Ries, B. 'Evaluation of Semantic Service Discovery - A Survey and Directions for Future Research', *Whitestein Series in Software Agent Technologies and Autonomic Computing*, Emerging Web Services Technology, ISBN: 978-3-7643-8863-8.

[157] Rambold, M., Kasinger, H., Lautenbacher, F. and Bauer, B. (2009) 'Towards Autonomic Service Discovery – A Survey and Comparison', *Proceedings of the IEEE International Conference on Services Computing*, Bangalore, India, pp. 192 - 201

[158] http://www.w3.org/TR/ws-arch/.

[159] Garofalakis, J., Panagis, Y., Sakkopoulos, E. and Tsakalidis, A. (2004) 'Web service discovery mechanisms: looking for a needle in a *haystack*' *Proceedings of the International Workshop on Web Engineering*, Munich, German, pp. 1-14.

[160] Cardoso, J. (2007) 'Semantic Web Services: Theory, Tools, and Applications', IGI Publishing Hershey, PA, USA, ISBN-13: 9781599040455

[161] Klein, M. and A. Bernstein, (2004) 'Toward high-precision service retrieval', *IEEE Internet Computing*, Vol. 8, No. 1, pp. 30-36.

[162] Le, D.N., Soong Goh, A.E. and Cao, T.H. (2007) 'A Survey of Web Service Discovery Systems' *International Journal of Information Technology and Web Engineering*, Vol. 2, No. 2, pp. 65-80.

[163] Benfenatki, H., Saouli, H., Benharkat, N., Ghodous, P., Kazar, O. and Amghar, Y. (2013) 'Cloud Automatic Software Development', *Proceedings of the 20th ISPE International Conference on Concurrent Engineering*, Melbourne, Australia, pp. 40-49.

[164] Brock, M. and Goscinski, A. (2009) 'Attributed Publication and Selection for Web Service-based Distributed Systems', *Proceedings of the World Conference on Services- I*, Los Angeles, USA, pp. 732- 739.

Erratum

- Concernant le chapitre V : On s'excuse du non netteté de certaines figures.
- On remercie d'avance toute personne qui nous signalera, les erreurs qu'il pourrait déceler, à l'adresse suivante : hamza_saouli@yahoo.fr