N° d'ordre : 19/2012- M/INF

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie « Houari Boumediene »

Faculté d'Electronique et d'Informatique

MEMOIRE

Présenté pour l'obtention du diplôme de MAGISTERE

En INFORMATIQUE

Spécialité : Informatique Mobile

Par : BOUSSOUALIM Nacera

Sujet

Etude et mise en œuvre d'un modèle de publication et de découverte des applications saas « software as a service »

Soutenu publiquement le 21 fev-2012, devant le jury composé de :

Mr M.AHMED-NACER, professeur à l'USTHB	Président
Mr-Y.AKLOUF, Maître de Conférence/A, à l'USTHB	Directeur de mémoire
Mr-Y.FEREDJ, Maître de Conférence/A, à l'USTHB	Examinateur
Mr-Y.BOUKHALFA, Maître de Conférence/A, à l'USTHB	Examinateur

Remerciements

A tous mes proches qui m'ont encouragé et qui ont cru en moi.

A tous ceux qui m'ont soutenu, qui m'ont supporté dans les bons et les moins bons moments durant ces trois années.

A tous ceux qui m'ont aidé de près ou de loin à réaliser ce travail, à avancer dans ma recherche.

A tous ceux qui ont pu contribuer à l'élaboration de ce travail :

- *A Monsieur Aklouf Youcef pour avoir accepté de diriger ce travail, pour sa confiance et ses encouragements.*
- *A Monsieur Ahmed Nacer, Monsieur feredj, et Monsieur Boukhalfa pour avoir accepté de faire partie de mon jury*
- *A toute la promotion de la PG 2008-2009, Option Informatique Mobile, en particulier Tarak et Atika pour leurs aides.*
- *A tout le personnel du département d'Informatique, je cite en particulier « Ami Aissa » du service de la Post Graduation pour ses aides et sa gentillesse.*
- *A ma sœur Aicha pour avoir lu et corriger ce mémoire de magistère.*

Résumé

Software as a Service (SaaS) ou autrement le logiciel en tant que service est la mise à disposition d'un logiciel non pas sous la forme d'un produit installé en interne, mais en tant qu'application accessible à distance comme un service, par le biais d'Internet et du Web. Les clients ne payent pas pour posséder le logiciel en lui-même mais plutôt pour l'utiliser. Cette nouvelle notion de (SaaS) est adoptée de plus en plus par les fournisseurs de logiciels d'une part et les entreprises d'autre part. Ce qui a entrainé une multiplication rapide du nombre d'applications SaaS disponibles sur le marché. Cette prolifération a donné naissance à plusieurs problèmes tels que l'intégration et la communication des données, la découverte et la sélection adaptée des produit SaaS les plus pertinents, la publication et la souscription des SaaS…etc. Dans ce mémoire, nous présentons un intergiciel qui permet à la fois l'intégration, la publication et la découverte des applications SaaS. Afin d'atteindre ce but, nous avons mis en œuvre une méthode de sélection d'un produit SaaS le plus pertinent ainsi qu'une méthode pour la génération d'un model de souscription aux Saas.

Abstract

Software as a Service (SaaS) is a software delivery paradigm in which the product is not installed on-premise, but it is available on Internet and Web. The customers do not pay to possess the software itself but rather to use it. Therefore the design of subscription management is critical for a SaaS provider. In the other hand, the number of services and users grow, so there is a need for a trusted service broker to manage services offered by service providers and subscribed customers. Thus, customers need to adopt an objective approach to ensure they select the most appropriate SaaS product for their needs. In this manuscript, we propose a trusted service broker for SaaS applications in which we integrate an approach that makes use of Analytic Hierarchy Process (AHP) technique for prioritizing the product features and also for expert-led scoring of the products, and a pattern-based approach for the subscription management design of SaaS.

Sommaire

Liste des figures

Liste des tableaux

INTRODUCTION GÉNÉRALE

Les statistiques ont montré que les ordinateurs personnels n'utilisent qu'entre 05 à 20% de leurs capacités, alors pourquoi on paye plus de ce qu'on utilise ? D'un autre côté, l'informatique est transformée en un modèle basé sur des services qui sont commandés et livrés d'une manière similaire aux autres services ou utilitaires traditionnels tels que l'eau, l'électricité, le gaz et la téléphonie. L'informatique, serait probablement un jour le 5^{éme} utilitaire ? Ce 5^{éme} utilitaire doit donc répondre aux besoins quotidiens de la communauté en fournissant un service fiable et accessible.

Pour réaliser cette vision, un certain nombre de paradigmes de l'informatique ont été proposés. Parmi ces derniers, on peut noter le grappe de serveurs « cluster Computing », les grilles de calcul «grid Computing » et le plus récent connu sous le nom de l'Informatique en nuage « Cloud Computing ». Dans ce dernier, le principe est que les données sont réparties sur un nuage de machines, les centaines de milliers d'ordinateurs-serveurs dont disposent les géants du web.

Cette vision est basée sur le service de l'informatique à la demande. Elle prévoit la transformation massive de l'industrie de l'informatique au XX^{le} siècle. Les applications et les ressources informatiques sont consommées en tant que services plutôt qu'ils sont installés chez les individus. Les consommateurs n'ont pas à investir lourdement dans la construction et la maintenance des infrastructures informatiques complexes. Dans un tel modèle, les utilisateurs accèdent aux services basés sur leurs exigences. Les traitements et les données désertent donc le micro-ordinateur familial pour rejoindre des centres distants auxquels les usagers accèdent à travers l'Internet à haut débit. Ces derniers sont surveillés et maintenus vingt-quatre heures sur vingt-quatre.

Les consommateurs, tels que les entreprises, sont attirés par cette nouvelle technologie dite le Cloud Computing pour la réduction des coûts liés à la fourniture interne « in-house » des services. Cependant, puisque les applications du Cloud Computing peuvent être cruciales aux opérations commerciales, il est essentiel que les consommateurs aient des garanties sur la livraison des services. Typiquement, ceux-ci sont fournis par des accords de niveau de service (SLAs) sponsorisés entre les fournisseurs et les consommateurs. En outre, les entreprises exigent un temps de réponse plus rapide, alors le Cloud Computing est un moyen pour économiser le temps en distribuant la charge de travail sur des nuages multiples dans divers endroits en même temps.

Amazone, Google, Salesforce, IBM, Microsoft et Sun Microsystems ont commencé à établir les nouveaux centres de calcul pour accueillir des applications de Cloud Computing dans divers endroits autour du monde pour assurer la fiabilité et l'accessibilité. Cependant, les exigences des utilisateurs changent fréquemment, alors les services doivent être flexibles et les utilisateurs doivent être indépendant des infrastructures (Buyya, Yeo, Venugo, Broberg, & Brandic, 2008).

Conçu pour accroître les avantages apportés par l'économie d'échelle, le SaaS (Sofware as a Service) est la troisième forme du Cloud. Il est la façon avec laquelle on doit fournir des fonctionnalités de logiciel à un grand groupe de clients à travers le web avec une instance unique d'application fonctionnant sur une plate-forme de multi-locataire (Chang, Wei, Ying, Zhi, & Bo, 2007). Les clients habituellement n'ont pas besoin d'acheter la licence et d'installer le progiciel dans leur environnement local. Ils emploient, seulement, les qualifications publiées par les fournisseurs de SaaS pour consommer le service à travers le web en utilisant un navigateur internet à tout moment et depuis n'importe quel endroit, il suffit d'avoir une connexion à internet.

PROBLEMATIQUE

Le logiciel en tant que service (SaaS) est adopté de plus en plus par les fournisseurs de logiciels d'une part et les entreprises d'autre part (Knorr, 2006). Bien que la notion du SaaS exploite la technologie

de services web comme la technologie de base, elle introduit des issues uniques, tels que le multi-tenancy ou bien le multi-client et la capacité de configuration du service SaaS pour chaque locataire. Ces issues n'ont pas été traitées par l'architecture orientée services (SOA) (Sun W. , Zhang, Guo, Sun, & Su, 2009) (Guo, Sun, Huang, Wang, & Gao, 2007) (Chong & Carraro, 2006). La gestion de l'abonnement est aussi un des issues qui doit être étudiée.

Si nous retournons au fameux ordinateur de bureau, qui est en réalité juste une interface à un système d'exploitation, les consommateurs oublient que c'est la colle qui non seulement héberge les applications mais aussi les surveille et permet à certaines entre elles de communiquer et échanger les données. Ainsi, nous perdons notre environnement cohérent à cause de la nature autonome des fournisseurs d'applications SaaS. Donc, il doit y avoir une sorte d'intergiciel indépendant pour que toutes les applications puissent communiquer. (Turner, Budgen, & Brereton, 2003).

De nombreux articles ont traité les applications SaaS, mais très peu se sont focalisés sur la façon d'intégrer ces derniers dans un environnement de confiance qui peut faciliter la communication sécurisée, la validation et l'échange de données entre les clients et les fournisseurs d'application. Le problème devient plus compliqué lorsqu'on pense à la communication et l'intégration de données entre les différentes applications SaaS.

Dans le secteur de logiciel d'entreprise, le logiciel est conçu pour supporter les opérations métier telles que : la gestion de données commerciales, l'automatisation et l'optimisation du processus métier. SaaS n'est pas un cas exceptionnel bien qu'il fournisse de multiples avantages. Parmi ces avantages, on trouve principalement que le coût total de la propriété du logiciel « Total Cost of Ownership(TCO) » est beaucoup plus réduit et que la mobilité est plus forte (Sheryl, 2004) (Jiehui, Ya, Jianqing, Jiyi, & Zhijie, 2010) (Boucher, 2009).

Cependant, chaque client est unique, ce qui mène à une très grande variation des exigences à propos du logiciel. Parmi les causes fondamentales de la variation des exigences entre les clients, on trouve : la différente focalisation de l'industrie, le différent comportement des clients, les différentes manières dont les produits sont offerts, les différents réglementaires, les différentes cultures et les différentes stratégies opérationnelles.

Quand plusieurs fournisseurs offrent des produits basés sur les SaaS, le choix du produit devient une question principale. Alors, pour prendre une décision sainte, une méthode de sélection basée sur multiple paramètres de choix doit être faite (Manish & Shrikant, 2009). Cela nécessite une définition et une analyse détaillé de ces paramètres de choix et des offres des produits des différents fournisseurs.

ORGANISATION DU MEMOIR

A travers notre travail de magistère, nous nous intéressons à ce domaine du SaaS et plus particulièrement à la construction d'une couche intermédiaire pour faciliter la découverte des applications SaaS les plus adaptées aux besoins du client et la souscription à ces dernières. Cette couche permettrait également aux fournisseurs de publier leurs applications SaaS. Elle leur donnerait la possibilité de bien décrire leurs produits afin d'être bien utilisés.

Afin de répondre à ces préoccupations et cette problématique, notre travail de recherche a suivi le cheminement suivant :

1. Nous avons commencé par un aperçu sur la notion du Cloud Computing et ses différentes formes.

2. Dans le chapitre 2, nous avons défini les applications SaaS en se concentrant particulièrement sur certains attributs permettant de distinguer une application SaaS des autres modèles d'applications.

3. Le troisième chapitre quand à lui, dresse une comparaison entre les SaaS et les types d'application hébergées ainsi que les différents avantages et les inconvénients de ce nouveau concept.

4. le quatrième chapitre survole quelques articles et travaux de recherche ayant trait au sujet de ce mémoire.

5. Dans le cinquième chapitre, nous avons proposé des solutions qui sont en mesure de résoudre quelques problèmes actuels des SaaS. Ces solutions touchent à la sélection d'un produit SaaS adapté aux besoins des clients et la définition d'un modèle de souscription aux applications SaaS

6. En dernier lieu, nous avons proposé une architecture détaillée de l'intergiciel proposé ainsi que quelques scenarios d'exemple.

PREMIER CHAPITRE : LE CLOUD COMPUTING

1 INTRODUCTION

John McCarthy a envisagé, en 1961, que « computer time-sharing technology might lead to a future in which computing power and even specific applications could be sold through the utility business model (like water or electricity)»[1]. Le Cloud Computing peut être considéré comme une grande étape vers ce rêve.

De nos jours, le Cloud Computing est devenu une réalité inévitable qui s'est imposée à plusieurs niveaux et dans pratiquement tous les domaines. Ainsi, l'informatique tend de plus en plus vers cette nouvelle technologie du Cloud Computing, ce qui a poussé beaucoup de grandes sociétés informatiques, connues sur l'échelle mondiale, à adopter cette nouvelle manière de voir.

Dans ce qui suit, nous faisons un aperçu de l'évolution historique du Cloud Computing : de son apparition à nos jours. Nous présentons, ensuite, les différentes formes du Cloud Computing ainsi que la différence entre lui et les autres paradigmes à savoir les Grappes de serveurs et les grilles de calcul à travers des définitions.

2 LE CLOUD COMPUTING : HISTORIQUE

Il est clair que d'une part, les évolutions successives des systèmes informatiques et l'ouverture des entreprises vers l'Internet, et d'autre part, le contexte actuel, à savoir la volonté de réduction des coûts, les nouveaux terminaux, etc. aboutissent logiquement au *Cloud Computing* (Plouin, 2009).

2.1 Le cycle des interfaces informatiques

Depuis sa montée en puissance dans les années 60, l'architecture informatique suit un cycle régulier de centralisation/décentralisation. Ainsi, les premiers systèmes utilisés en entreprises étaient des mainframes. Après, au début des années 90 sont apparues les architectures client/serveur qui ont permis le report des traitements sur les postes de travail, les fameux ordinateurs personnels ou PC, inventés par IBM. L'idée novatrice du client/serveur était de répartir les traitements entre un serveur et un poste utilisateur qui est devenu capable d'exécuter certains processus métier. L'architecture client/serveur a été massivement utilisée, mais elle a fini par montrer ses limites. En effet, l'absence de standardisation du protocole d'échange rendait difficile la gestion des flux.

Au milieu des années 1990, les architectures web ont conduit à la recentralisation de la logique de traitement sur des serveurs centraux, ramenant le PC à un simple dispositif d'affichage au travers un navigateur. Elles ont permis l'usage d'applications à l'échelle de l'Internet grâce aux standards *http* « *HyperText Transfer Protocol* » et HTML « *HyperText Markup Language* »[2]. De plus, elles ont permis un accès aux applications sans passer par la douloureuse phase de déploiement logiciel sur chacun des PC du parc informatique.

2.2 La montée en puissance du web

Lorsque le web est devenu une plate-forme mondiale, les entreprises l'ont utilisée pour diffuser des plaquettes commerciales à moindre coût : les fameux « sites vitrines ». Puis à la fin des années 1990,

1 http://ComputingintheCloud.wordpress.com/2008/09/25/utility-Cloud-Computingflashback-to-1961-prof-john-mccarthy/
2 Les standards du web (HTTP et HTML) ont été inventés en 1990 par Tim Berners-Lee. Son idée initiale était de créer une sorte d'encyclopédie en ligne.

ces sites ont commencé à devenir transactionnels, permettant l'émergence du commerce électronique, pour devenir de véritables applications informatiques.

Le web a aussi introduit un changement dans l'évolution de l'informatique : en effet, des innovations ont commencé à être testées auprès du grand public (par exemple les moteurs de recherche), avant d'être déclinées pour les entreprises (Plouin, 2009).

2.3 L'émergence de l'ASP « Application Services Providers »

C'est à cette période qu'est né le concept des ASP[3], les *Application Services Providers*. Des créateurs de start-up ont vu le parti qu'ils pouvaient tirer des architectures web : proposer aux entreprises de louer des applications métiers hébergées par leurs soins, dans leurs centres serveurs. Les ASP promettaient à leurs éditeurs des revenus réguliers grâce à un système d'abonnement. Elles promettaient aux entreprises utilisatrices de se débarrasser des problématiques d'exploitation de ces applications.

À cette époque, deux alternatives s'offraient aux applications en ASP :

1. **Utiliser une interface web** : À ce stade, il est important de souligner qu'accéder à une application de collaboration ou à une application métier depuis une interface HTML élémentaire peut se révéler très frustrant. En effet, ces dernières proposent une navigation de page en page suivant un scénario préétabli. Ce mode d'interaction est inadapté à une application ASP destinée à un usage quotidien.

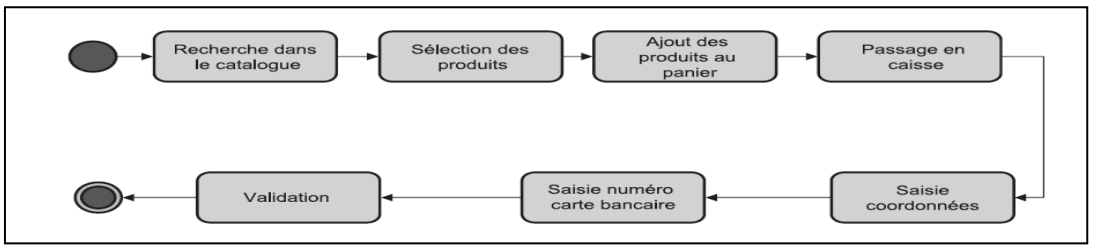


Figure 1. La navigation web et la contrainte d'un scénario préétabli

2. **Utiliser une interface client/serveur :** Ce type d'interface est beaucoup plus satisfaisant en termes d'ergonomie. Cependant, il nécessite un déploiement sur les postes utilisateurs, ce qui va à l'encontre de la promesse des ASP : fournir une application en mode hébergé. En effet, on retombe là dans la fameuse problématique de déploiement propre aux applications internes à l'entreprise. De plus les middlewares utilisés par les applications client/serveur sont souvent bloqués par les firewalls d'entreprise, ce qui complexifie beaucoup leur déploiement. Cette problématique d'interface est la principale raison de l'échec des ASP. Nous verrons dans la suite que les interfaces RIA ont résolue ce problème dans le cadre du Cloud Computing.

Sur le plan technique, les applications en ASP s'appuyaient sur des architectures similaires à celles des applications d'entreprises classiques et reposaient ainsi sur :

- Une application unique ;
- Une version unique de l'application ;
- Une base de données unique ;

3 http://www.ei-technologies.com/detail.do?noArticle=120&idRubrique=48;
http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci213801,00.html

- Un système d'authentification unique.

Or une application ASP ne s'adresse pas à une communauté d'utilisateurs unique, mais à « N » communautés correspondant à « N » entreprises. Cela peut induire une forte volumétrie de données, difficile à gérer avec une base unique (Plouin, 2009).

D'autre part, il peut être pertinent, pour des raisons de sécurité, de séparer les données des différentes entreprises, ainsi que leur système d'authentification. De plus, une entreprise cliente pourra souhaiter customiser son application afin d'intégrer des spécificités propres à son métier (Plouin, 2009).

2.4 Les RIA[4] : une nouvelle opportunité pour les applications hébergées

Le concept du « client riche » est né en 2003. Il désigne une interface à la croisée des chemins entre les mondes client/serveur (ou client lourd) et web ou (client léger). Le qualificatif « riche » désigne sa capacité à être enrichi par rapport au client léger. Le concept du client riche a été affiné par la suite et il est aujourd'hui divisé en deux sous-catégories :

- Le RIA, *Rich Internet Application*, client riche basé sur un navigateur et successeur des applications web.
- Le RDA, *Rich Desktop Application*, client riche installé sur le poste de travail et successeur des applications client/serveur. il est déployé et mise à jour sur http et il sait gérer le mode déconnecté grâce à un stockage local.

Le RIA fonctionne alors comme une application client/serveur, le client étant l'interface RIA. Cette dernière persiste au sein du navigateur pendant toute la durée d'usage de l'application. Elle disparaît du poste utilisateur à la fermeture du navigateur.

Le RIA constitue donc une certaine forme de retour à une architecture Client/ serveur, mais sans problématique de déploiement sur les postes de travail. Le défaut majeur du RIA est l'absence de gestion du mode déconnecté (travail dans un train ou un avion). Lorsqu'on perd le réseau ou lorsqu'on ferme le navigateur, tout est perdu (Plouin, 2009).

Les technologies RIA disponibles aujourd'hui sont :

1. **Ajax[5], basé sur le standard JavaScript ;** l'acronyme Ajax signifie *Asynchronous JavaScript and XML*. Il permet la création d'interfaces métiers ou d'interfaces grand public très dynamiques. Par contre, Ajax n'a pas de capacités multimédias.
2. **Adobe Flash ;** crée en 1996, la technologie Flash a initialement été conçue pour permettre la création d'animations vectorielles au sein de pages web. Flash fonctionne avec un plug-in, une extension gratuite à installer en complément d'un navigateur web. Flash a permit l'intégration d'images, de sons et de la vidéo. Aujourd'hui, Flash permet la création d'interfaces métiers événementielles, en remplacement des interfaces client/serveur et la création d'interfaces multimédia.
3. **Microsoft Silverlight ;** Silverlight a été créé par Microsoft en 2007 afin de compléter son offre de technologies d'interface et d'offrir une alternative maison à Adobe Flash. En effet, Microsoft ne

4 http://blog.aysoon.com/Que-sont-les-applications-riches-RIA-Partie-1-Definition-et-usages-195
5 http://ajaxmar08.sys-con.com/node/609607 ; http://www.asp.net/ajax/ ; http://ajaxoct07.sys-con.com/

disposait jusqu'alors d'aucune véritable technologie RIA. Silverlight se présente comme un plug-in multi-navigateur concurrent frontal de Flash.

Ajax a l'avantage d'être entièrement basé sur des standards. Les deux autres technologies sont propriétaires mais elles offrent des effets de transparence et du multimédia (audio/vidéo).

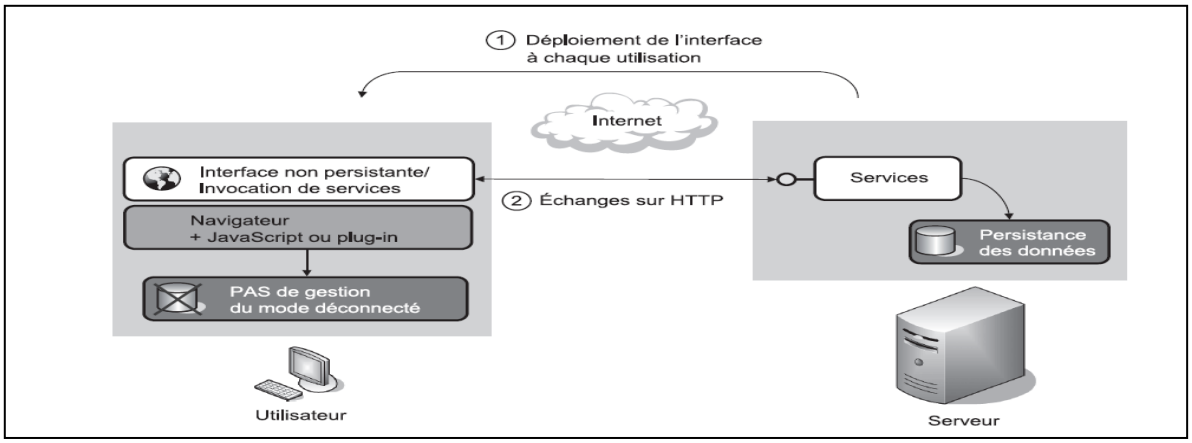


Figure 2. Le fonctionnement des RIA

Le RIA met fin au choix cornélien entre application web et application client/ serveur. Il offre, en effet, une solution purement web, sans problématique de déploiement, tout en bénéficiant d'une architecture client/serveur décentralisée : une interface ergonomique, agile, permettant une bonne productivité. Le RIA offre donc une solution très pertinente aux problématiques d'interface des ASP. C'est une des briques fondamentales à l'émergence du Cloud *Computing* (Plouin, 2009).

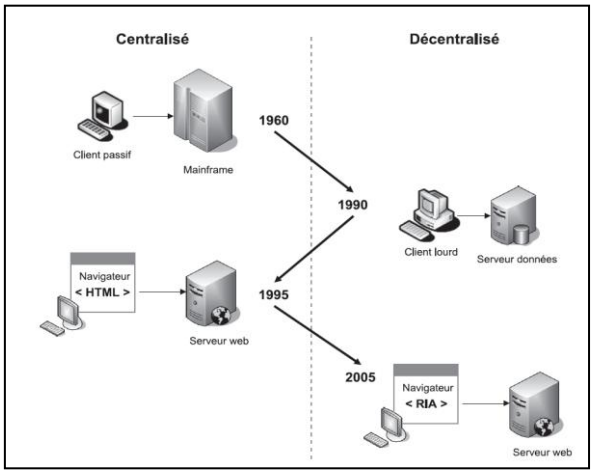


Figure 3. Le RIA, le meilleur des mondes web et client/serveur

La seule limite du RIA est la non-gestion du mode déconnecté, mais cette limite est en phase de disparaître en utilisant par exemple une nouvelle génération de navigateur gérant le mode déconnecté et des logiciels de synchronisation (Plouin, 2009).

2.5 Le web 2.0 : une nouvelle pertinence aux applications hébergées

Apparu en 2005, le terme web 2.0[6] est difficile à définir. Il recouvre une nébuleuse de nouveaux usages et de nouveaux outils, que nous allons essayer de décrire dans ce paragraphe. Nous allons essayer de donner une définition « pragmatique » du web 2.0.

2.5.1 L'intelligence collective et les digital natives

Le web 2.0 repose avant tout sur le concept d'«intelligence collective » ou « sagesse des foules ». Ces termes désignent les synergies qui peuvent avoir lieu entre des individus qui rédigent des textes sur le web afin de bâtir une somme de connaissances. Le meilleur exemple d'intelligence collective est l'encyclopédie en ligne « Wikipedia ». Cependant, dans son ouvrage «Comment le web change le monde », Francis Pisani pense que les synergies entre les contributions ne sont pas nécessairement constructives.

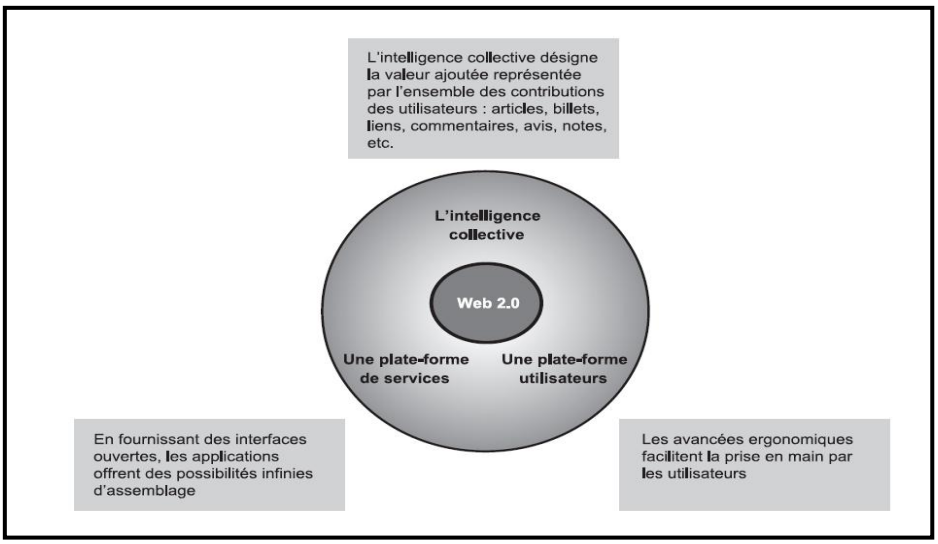


Figure 4. Les concepts du web 2.0

Les plus grands contributeurs à cette intelligence sont issus de la jeune génération, les fameux «digital natives » ou « génération Y » pour qui l'usage de l'Internet est complètement naturel. Ces utilisateurs ont une telle habitude des espaces collaboratifs en ligne qu'ils vont naturellement pousser leur entreprise à utiliser des outils similaires, disponibles sous forme SaaS. Ils seront donc les promoteurs des SaaS.

Les outils associés à l'intelligence collective sont les blogs, les wiki, et plus largement les sites web qui incitent à la participation. Ils ont largement contribué à l'usage d'applications web hébergées.

2.5.2 Une plate-forme utilisateurs

Le mouvement du web 2.0 s'est bâti sur les applications web plus ergonomiques, plus faciles à utiliser que les applications des générations précédentes. Cette ergonomie a été rendue possible par les technologies RIA introduites dans les paragraphes précédents. Un des meilleurs exemples de cette amélioration ergonomique est la vidéo en ligne. Dans le passé, diffuser une vidéo sur Internet

6 http://sites.google.com/site/leweb20/ ; http://fr.wikipedia.org/wiki/Web_2.0

nécessitait des compétences pointues. Aujourd'hui, des plates-formes comme Youtube rendent cette mise en ligne extrêmement simple. Un autre exemple plutôt impressionnant est Google Spreadsheet: ce service offre des fonctions de tableur assez proches de celle de Microsoft Excel.

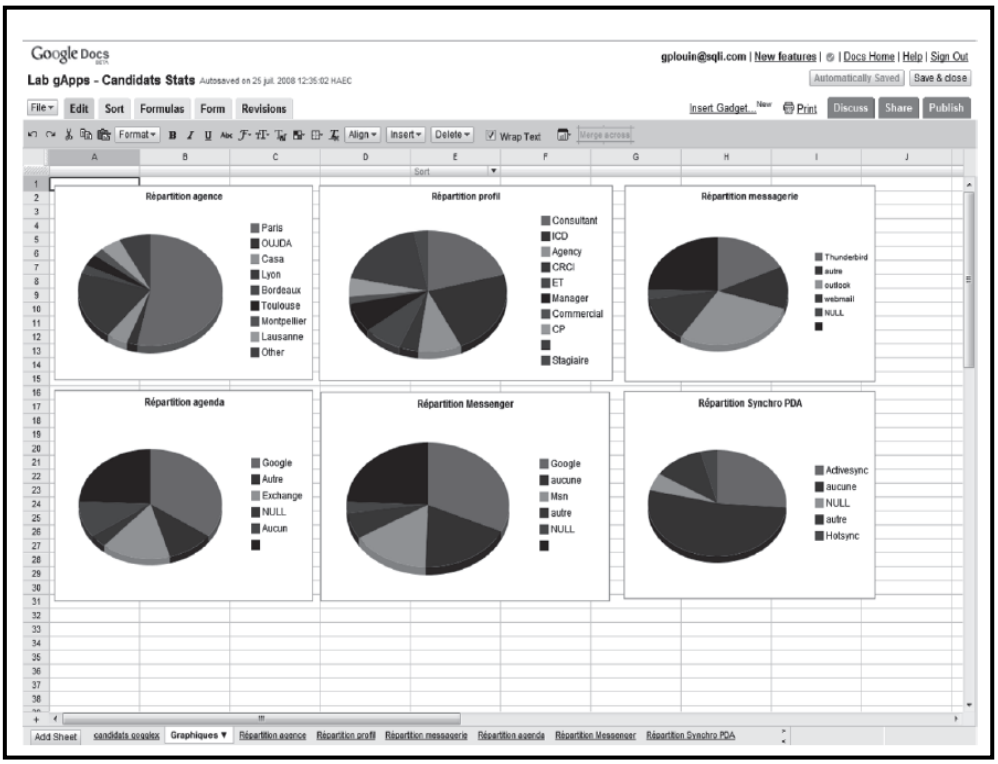


Figure 5. Le tableur Google Spreadsheet

Ces applications orientées grand public, et souvent gratuites, ont développé une ergonomie telle que leur usage est devenu quotidien pour leurs utilisateurs, les « digital natives ».

Certaines applications web 2.0 ont pris une telle importance dans le quotidien de leurs utilisateurs, qu'elles sont devenues pour eux des applications critiques. Elles ont donc dû assurer une qualité de service irréprochable, et les plates-formes techniques des grands acteurs du web, comme Amazon, Yahoo ou Google, sont devenues des modèles de performance et de robustesse. Les centres de données bâtis par ces acteurs sont à l'origine des architectures des PaaS (*Platform as a Service*) sous-jacentes aux SaaS (Plouin, 2009).

2.5.3 Une plate-forme de services

La plupart des applications du monde web 2.0 mettent à disposition des API (*Application Programming Interface*), sortes d'interfaces qui permettent l'invocation de leurs services depuis d'autres applications. Ces API sont ouvertes, publiques et utilisables par tous. Il est donc possible de créer des applications qui recourent à ces services.

Lorsque des applications sont bâties uniquement sur la base de ces API, on les appelle applications composites «mashups ». Elles sont construites par assemblage libre, à la manière des « legos ».

Le site housingmaps.com est un des exemples les plus connu de mashup : il fait appel à l'API de Craiglist, un site de petites annonces, et à l'API de Google Maps, une solution de cartographie en ligne. La résultante de cette application composite est une carte des petites annonces immobilière que l'on peut parcourir et agrandir selon ses besoins (Plouin, 2009).

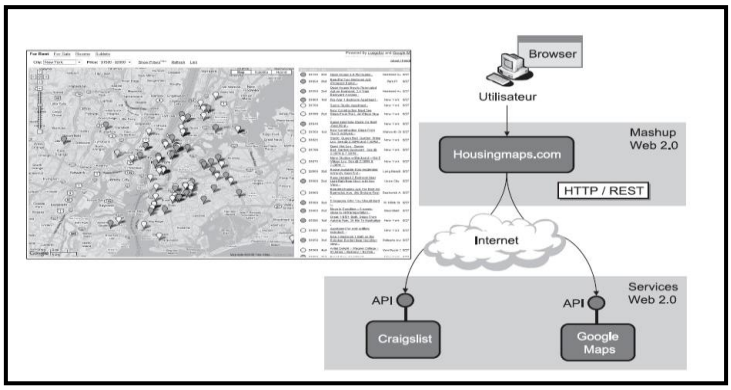


Figure 6. Le principe des mashups avec l'exemple d'housingmaps

Ce principe d'API offre un grand potentiel créatif aux acteurs du web 2.0. En effet, ils peuvent construire une myriade d'applications par combinaison d'interfaces. Le site programmableweb.com référence d'ailleurs plus de 3 300 *mashups* à la fin de l'année 2008 (Plouin, 2009).

2.6 Renforcement de la pertinence des applications hébergées par les nouveaux terminaux

Le propos de ce paragraphe est de montrer que des interfaces alternatives émergent, et qu'elles renforcent la pertinence des applications hébergées. Depuis l'apparition du Palm Pilot en 1996, les appareils mobiles ont beaucoup évolué. Ils ont connu diverses appellations comme PDA (*Personal Digital Assistant*), Smartphones, PDAphones, etc. Ils se sont dotés de capacités de communication de plus en plus sophistiquées. Par conséquent, il est très difficile d'utiliser des applications client lourd sur ces appareils : elles sont non seulement complexes à créer, mais en plus il faudrait en développer des centaines de variantes pour adresser tous les appareils mobiles du marché. C'est l'une des raisons pour lesquelles on privilégie les applications hébergées pour les appareils mobiles.

Le mode hébergé assure aussi l'intégrité des données, mal protégées par les petits appareils, sujets à des vols, à de la casse, ou à des pannes de batteries entraînant la perte de données. Par ailleurs, les utilisateurs d'appareils mobiles utilisent généralement en parallèle un PC classique, et ces deux appareils doivent accéder aux mêmes informations. Cet accès concurrent est grandement facilité lorsque les applications et les données sont sur le web. La montée en puissance des appareils mobiles renforce donc la pertinence des applications hébergées, ou SaaS (Plouin, 2009).

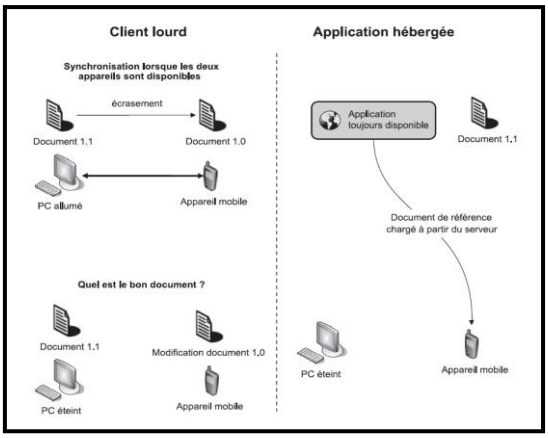
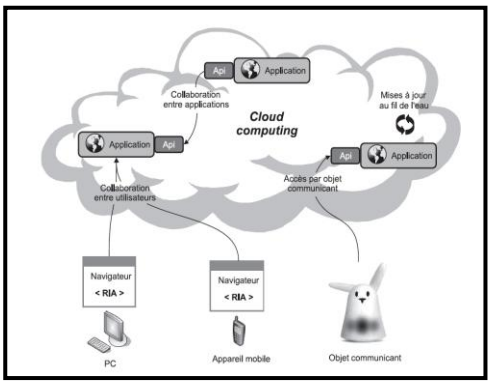


Figure 7. Pertinence d'une application hébergée en cas d'accès par plusieurs appareils

2.7 Le Cloud Computing : capitalisation de toutes les évolutions précédentes

En proposant l'hébergement des applications sur des plates-formes accessibles depuis le web, le *Cloud Computing* est l'aboutissement de l'ensemble des mouvements évoqués dans ce que nous venons de situer (voir figure ci-dessous).



Le *Cloud Computing* va bien au-delà de la synthèse des mouvements précédents. Le *Cloud Computing* a tiré les leçons de l'échec de l'ASP en proposant des architectures plus adaptées à la consommation d'applications depuis le web. Il a intégré les pratiques issues du web 2.0 comme la collaboration et la construction d'applications par assemblage d'API. Nous pouvons également noter que le web 2.0 a préparé les utilisateurs « *early adopters* » à utiliser des applications hébergées. Enfin la montée en puissance des appareils communicants de toutes sortes rendent le modèle des applications hébergées incontournable. Nous présentons ses autres bénéfices dans les parties suivantes.

3 LE CLOUD COMPUTING : DEFINITIONS, CARACTERISTIQUES ET FORMES

Les entreprises disposent aujourd'hui d'offres leur permettant de se soucier le moins possible de l'administration de leur informatique. Elles peuvent ainsi se concentrer sur leur métier, c'est-à-dire sur ce qui fait leur valeur ajoutée. Le Cloud Computing a pour but d'offrir des services qui vont au-delà des offres classiques et que nous pouvons tenter de définir avec les caractéristiques suivantes :

- Ils s'agit d'une informatique distribuée où les échanges sont gérés et centralisés par des serveurs distants, les applications étant stockées non plus sur le poste de travail, mais sur un "nuage" (Cloud) de serveurs, accédées par une connexion Internet et un navigateur web.
- Les applications, plateformes et infrastructures nécessaires sont louées en fonction de l'usage qui en est fait, que ce soit pendant le développement de ces applications ou pendant leurs utilisations en production.
- Les applications, plateformes et infrastructures sont facilement extensibles.
- Les ressources peuvent être allouées dynamiquement en fonction du besoin.
- Les applications, plateformes et infrastructures restent disponibles en cas de panne d'une ressource.

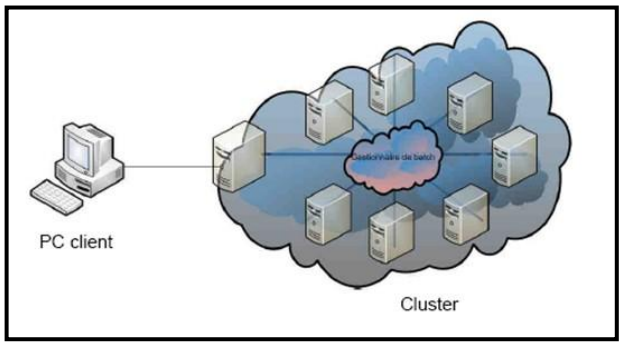
Afin de construire une idée claire de ce qui est exactement le Cloud Computing, nous allons le comparer à deux autres paradigmes récents, largement adoptés et explorés en informatique qui sont : le Grappe de serveurs[7] « cluster Computing » (Pfister, 2003) et les grilles de calcul «grid Computing ». Nous définissons d'abord ces trois paradigmes ainsi que leurs caractéristiques. Nous présentons en dernier les différentes formes du Cloud.

7 http://fr.wikipedia.org/wiki/Grappe_de_serveurs

3.1 Quelques définitions

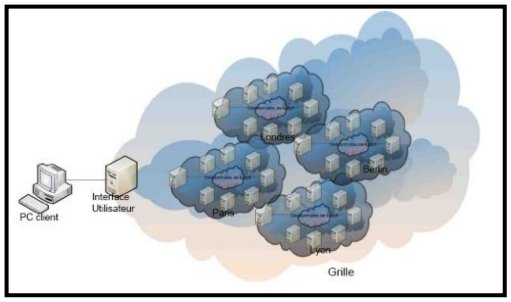
Un certain nombre de chercheurs en informatique ont essayé de définir les trois paradigmes qui sont : les Grappes de serveurs « cluster Computing », les grilles de calcul «grid Computing » et l'informatique en nuage « Cloud Computing » de diverses manières. Voici quelques définitions assez génériques :

Une grappe de PC, ou tout simplement un *cluster,* est un type de système parallèle réparti. Plus précisément, c'est une collection d'ordinateurs connectés entre eux par un réseau et utilisés comme une ressource informatique unique.



Les grappes ont été utilisées comme un moyen simple pour obtenir une plus grande capacité et une meilleure fiabilité qu'on ne peut pas l'acquérir en utilisant un ordinateur simple. Les clusters peuvent être un cadre informel, voire anarchique, de l'organisation informatique, ils n'ont pas été construits par des fabricants d'ordinateurs mais plutôt assemblés par les clients sur une base *ad hoc* (Pfister, 2003) (Ralston, Reilly, & Hemmendinger, 2003)((Buyya, Yeo, Venugo, Broberg, & Brandic, 2008).

(Abramson, Buyya, & Giddy, 2002), (Buyya & Venugopal, 2005) et (Buyya, Yeo, Venugo, Broberg, & Brandic, 2008) définissent la grille comme un type de système parallèle et distribué qui permet le partage, la sélection, et l'agrégation de ressources autonomes géographiquement distribués dynamiquement. Chacune de ces ressources ont leur propre disponibilité, capacité, performance, coût et utilisabilité, avec leurs propres contraintes de qualité de service.



Dans leur article "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility" apparu en 2008, R. Buyya et al. définissent le Cloud Computing ainsi : "Le Cloud Computing est un type de système parallèle et distribué qui se compose d'une collection d'ordinateurs virtuels et interconnectés présentés en tant qu'un ou plusieurs ressources unifiées pour répondre au services exigés. Typiquement, ceux-ci sont fournis par des accords de niveau de service (SLAs) établis par la négociation entre le fournisseur de service et les consommateurs ». Ainsi, les Cloud semblent être une combinaison des grappes et des grilles. Alors que, ce n'est pas le cas. Les Clouds sont clairement une nouvelle génération des data-centres dynamiquement approvisionné.

3.2 Les caractéristiques des trois paradigmes

Un ensemble de caractéristiques des trois paradigmes que nous venons de définir « les grappes, les grille et les Clouds » est présenté dans le tableau n° 1 pour pouvoir les distingué les uns des autres.

Systems / Characteristics	Clusters	Grids	Clouds
Population	Commodity computers	High-end computers (servers, clusters)	Commodity computers and high-end servers and network attached storage
Size / Scalability	100s	1000s	100s to 1000s
Node Operating System (OS)	One of the standard OSs (Linux, Windows)	Any standard OS (dominated by Unix)	A hypervisor (VM) on which multiple OSs run
Ownership	Single	Multiple	Single
Interconnection Network | Speed	Dedicated, high-end with low latency and high bandwidth	Mostly Internet with high latency and low bandwidth	Dedicated, high-end with low latency and high bandwidth
Security/Privacy	Traditional login/password-based. Medium level of privacy – depends on user privileges.	Public/private key pair based authentication and mapping a user to an account. Limited support for privacy.	Each user/application is provided with a virtual machine. High security/privacy is guaranteed. Support for setting per-file access control list (ACL).
Discovery	Membership services	Centralised indexing and decentralised info services	Membership services
Service Negotiation	Limited	Yes, SLA based	Yes, SLA based
User Management	Centralised	Decentralised and also Virtual Organization (VO)-based	Centralised or can be delegated to third party
Resource Management	Centralized	Distributed	Centralized/Distributed
Allocation / Scheduling	Centralised	Decentralised	Both centralised/decentralised
Standards / Inter-Operability	Virtual Interface Architecture (VIA)-based	Some Open Grid Forum standards	Web Services (SOAP and REST)
Single System Image	Yes	No	Yes, but optional
Capacity	Stable and guaranteed	Varies, but high	Provisioned on demand
Failure Management (Self-healing)	Limited (often failed tasks/applications are restarted).	Limited (often failed tasks/applications are restarted).	Strong support for failover and content replication. VMs can be easily migrated from one node to other.
Pricing of Services	Limited, not open market	Dominated by public good or privately assigned	Utility pricing, discounted for larger customers
Internetworking	Multi-clustering within an Organization	Limited adoption, but being explored through research efforts such as Gridbus InterGrid	High potential, third party solution providers can loosely tie together services of different Clouds
Application Drivers	Science, business, enterprise computing, data centers	Collaborative scientific and high throughput computing applications	Dynamically provisioned legacy and web applications, Content delivery
Potential for Building 3rd Party or Value-added Solutions	Limited due to rigid architecture	Limited due to strong orientation for scientific computing	High potential – can create new services by dynamically provisioning of compute, storage, and application services and offer as their own isolated or composite Cloud services to users

Tableau 1. Les principaux Caractéristiques des grappes, des grilles et des Clouds. (Buyya, Yeo, Venugo, Broberg, & Brandic, 2008)

Dans les grappes, les ressources sont situées dans un Domain Administratif simple et contrôlées par une entité simple tandis que, dans les grilles, des ressources sont géographiquement distribuées à travers des domaines administratifs multiples tels que chacun d'eux a sa propre politique de gestion et ses propres buts.

Une autre différence principale entre les grappes et les grilles résulte de la manière d'effectuer l'ordonnancement des applications. Les ordonnanceurs dans les grappes se concentrent sur l'amélioration des performances et l'utilité de système global car ils sont responsables du système entier. Alors que les ordonnanceurs dans les grille sont appelés ressources Brokers, ils se concentrent sur amélioration de la performance des applications spécifiques de telle manière que les exigences de ses utilisateurs en ce qui concerne la QoS soient satisfaites.

Les plateformes du Cloud possèdent des caractéristiques héritées des grappes et des grilles à la fois, avec ses propres attributs tel que : une forte virtualisation, une composition dynamique du service en utilisant les interfaces du Web service, le stockage…etc. Ainsi, le Clouds Computing fournit des services aux utilisateurs sans être lier à l'infrastructure sur laquelle ceux-ci sont accueillis.

3.3 Les formes du Cloud Computing

Plus précisément, nous distinguons trois formes de Cloud Computing :

- SaaS « Softwar as a Service » ;
- PaaS « Platform as a Service » et
- IaaS « Infrastructure as a Service ».

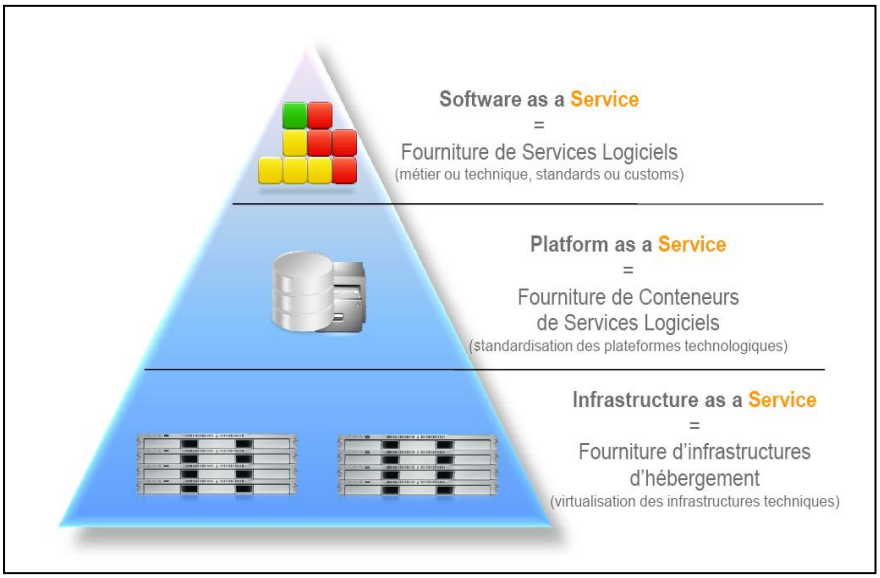


Figure 8. Les formes du Cloud Computing (Macquet, 2009)

3.3.1 SaaS (Software as a Service)

Il s'agit de la mise à disposition d'un logiciel non pas sous la forme d'un produit que le client installe en interne, mais en tant qu'application accessible à distance, par le biais d'Internet. Les clients ne payent pas pour posséder le logiciel en lui-même mais plutôt pour l'utiliser. Ils l'utilisent soit directement via l'interface disponible, soit via des API fournies (souvent réalisées grâce aux Web Services ou à l'architecture **REST**[8] (**Representational State Transfer**)). L'utilisation reste transparente pour les utilisateurs, qui ne ce soucient ni de la plateforme, ni du matériel qui sont mutualisés avec d'autres entreprises.

Le SaaS est dès à présent une solution intéressante pour les PME « petite et moyenne entreprise » et les TPE « très petite entreprise », et devient également attractif pour les grandes entreprises. De plus en plus, les éditeurs classiques commencent à proposer également des versions en ligne de leurs produits sous forme de location.

8 http://fr.wikipedia.org/wiki/Representational_State_Transfer

Les principales applications actuelles de ce modèle sont la relation client (Customer RelationShip Management CRM), la vidéo conférence, la gestion des ressources humaines, les communications unifiées, le travail collaboratif, les emails. Nous retrouverons dans cette catégorie comme premiers et principaux acteurs Salesforce.com (logiciels CRM) et Google (Gmail, Google Apps).

3.3.2 PaaS (Platform as a Service)

Il s'agit de la mise à disposition, pour une entreprise, d'environnements techniques pour le développement des applications fonctionnant à distance comme pour les SaaS mais en incluant des outils de personnalisation et une intégration à l'existant ou à d'autres programmes hébergés.

L'objectif est ainsi de proposer un environnement modulaire capable de combiner plusieurs fonctions et processus métier, voire plusieurs technologies en provenance de divers éditeurs. Il faut noter cependant qu'il n'y a pas de standard PaaS. Un logiciel développé sur une plateforme ne fonctionnera pas actuellement sur une autre plateforme.

La plateforme est conçue pour que l'approvisionnement de matériel et la montée en charge soient transparents. Le *PaaS* offre ainsi une grande flexibilité, permettant notamment de tester rapidement un prototype ou encore d'assurer un service informatique sur une période de courte durée. Il favorise également la mobilité des utilisateurs puisque l'accès aux données et aux applications peut se faire à partir de n'importe quel périphérique connecté.

Les principaux acteurs sont ici SalesForce.com (Force.com), Google (Google App Engine), Microsoft (Windows Azure), Facebook (Facebook Platform).

3.3.3 IaaS (Infrastructure as a Service)

Il s'agit de la mise à disposition, à la demande, de ressources d'infrastructures dont la plus grande partie est localisée à distance dans des Datacenters. Les serveurs, postes de travail, et imprimantes peuvent être facturés en fonction de leur utilisation. Le client peut louer n'importe quel ressource informatique par exemple le CPU, la RAM ou le disk pour le stockage de données et le coût est directement lié au taux d'occupation. Une analogie peut être faîte avec le mode d'utilisation des industries des commodités (électricité, eau, gaz) ou des Télécommunications.

Les principaux acteurs de l'IaaS sont actuellement : Amazon (EC2/S3)[9] et IBM (Bluehouse)[10].

Il est à noter également dans la littérature informatique d'autres termes utilisés dans le cadre du Cloud Computing qui peuvent ajouter un peu plus de confusion sur le sujet :

- HaaS *(Hardware as a Service)*.peut être considéré comme un équivalent à IaaS
- DaaS *(Database as a Service*) concerne plus précisément les bases de données et peut s'apparenter à une composante de PaaS.
- Development as a service peut se comprendre comme un environnement de développement et de test intégré à la plateforme PaaS.
- Integration as a service est un bus d'échange intégré à la plateforme PaaS

9 http://aws.amazon.com/ec2 ; http://aws.amazon.com/s3
10 http://www.silicon.fr/fr/news/2008/08/25/ibm__lotus_foundation_et_le_saas_bluehouse_

4 CONCLUSION

Le Cloud Computing est donc la convergence de ses différentes solutions (SaaS, PaaS, IaaS). De nombreuses offres hétérogènes dans toutes ces catégories de Cloud Computing sont disponibles ou vont l'être dans le futur proche.

Dans notre travail de magistère, nous nous intéressons particulièrement au SaaS. C'est la raison pour laquelle, nous développons dans la partie suivante les applications SaaS.

DEUXIEME CHAPITRE : LES SAAS : MODELE METIER, ARCHITECTURE ET STRUCTURE OPERATIONNEL

1 INTRODUCTION

En 2003, M. Turner et al, ont avancé que le concept de base des SaaS est la dissociation de la propriété du logiciel et les droits d'utilisation de ce dernier (Turner, Budgen, & Brereton, 2003) (Liao & Tao, 2008). Avec l'apparition et le soutien du Web2.0, en juin 2006, *Salesforce* a publié son produit *online-CRM* basé sur le modèle SaaS et la théorie de la longue traîne de Chris Anderson (Anderson, 2006). Dans la même période, de nombreux fournisseurs de logiciels traditionnels ont commencé à migrer vers le domaine de SaaS. En novembre 2006, Microsoft a introduit des logiciels appelés « *Live online service »*. En juillet 2007, Oracle Corporation a annoncé la version la plus récente de son logiciel CRM hébergé *Oracle Siebel*. SAP, le plus grand fabricant de logiciels à l'époque, a lancé en septembre 2007, le logiciel de gestion d'entreprise SaaS appelé *A1S*.

La croissance des besoins des logiciels d'une part et la préoccupation du coût d'utilisation de logiciel et les bénéfices économique d'une autre part ont stimulé l'intérêt accru de l'utilisation du mode SaaS. Qu'est-ce que le SaaS ou encore le logiciel en tant que service ? Seulement quelques personnes ont pu vraiment le définir, et très peu sont ceux qui savent comment le construire. Nous pensons que le SaaS aurait un impact majeur sur l'industrie du logiciel, il changerait éventuellement la façon dont on construit, vend, achète et utilise le logiciel.

2 DEFINITION DU SAAS

Aujourd'hui encore, la définition exacte du SaaS est ouverte au débat. Néanmoins, la plupart des experts se mettent d'accord sur quelques principes fondamentaux qui distinguent les SaaS des progiciels traditionnels d'une part et des sites Web simples d'autre part.

Le SaaS peut alors être caractérisé tout simplement comme un « logiciel déployé en tant que service hébergé et accessible à travers Internet » (Chong & Carraro, 2006). En examinant les implications de cette définition, les remarques suivantes peuvent êtres éditées :

- Elle ne prescrit pas une architecture d'application spécifique ;
- Elle ne spécifie pas une technologie ou un protocole particulier ;
- Elle n'établit pas une distinction entre services orienté métier et les services orientée consommateurs, et
- Elle n'exige pas des modèles d'entreprise spécifiques.

Selon cette définition, les fonctions clés distinctives du SaaS sont :

- Où réside le code de l'application, et
- Comment les SaaS sont déployés et accessible.

Si on s'appuie sur cette définition, un certain nombre de services et d'applications peuvent se trouver dans la catégorie des SaaS alors qu'ils ne le sont pas en réalité. Nous prenons à titre d'exemple, les services de messagerie Web, tels que Microsoft Hotmail. Bien que le Hotmail ne puisse pas être une application SaaS, elle répond à tous les critères de base de la définition précédente : un fournisseur héberge tous : la logique de la programmation et les bases de données et permet, aux utilisateurs finaux, d'accéder à ces données à travers Internet, par le biais d'une interface utilisateur basée sur le Web.

Cette définition est donc un peu simple. Nous nous concentrerons, dans une autre partie, sur certains attributs qui définissent et distinguent une application SaaS des autres applications.

Nous pouvons identifier deux grandes catégories de logiciels SaaS en passant du général au spécifiques :

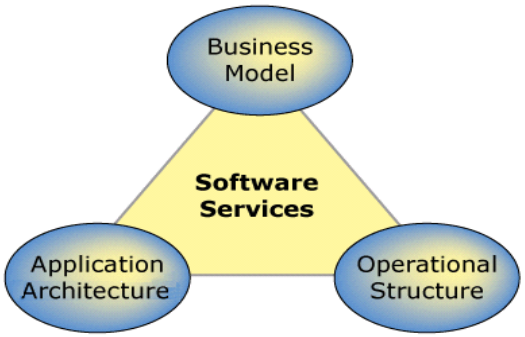
1. **Des services du ligne-de-business « *Line-of-business service* » :** offerts aux entreprises et aux organisations de toutes tailles. Les services de cette catégorie sont souvent des grandes solutions, personnalisables, qui visent à faciliter les processus d'entreprise comme par exemple la gestion de la chaîne d'approvisionnement, les finances, la comptabilité et la relations avec la clientèle. Le payement de ce genre de services se fait typiquement par abonnement. (Chong & Carraro, 2006) (Weier & Smith, 2007) (Boucher, 2009).
2. **Des services orientée-consommateurs « Consumer-oriented services » :** offerts au grand public. Les services de cette catégorie sont parfois payés par abonnement, mais sont souvent fournis aux consommateurs gratuitement. Cela rentre dans le cadre de la publicité. (Liao & Tao, 2008) (Chong & Carraro, 2006).

Dans la partie suivante, nous nous intéresserons particulièrement à l'architecture impliquée dans le développement des applications SaaS de type « line de business ». Nous aborderons quelques issus des applications développées sous ce nouveau paradigme telles que la personnalisation, le multi-client, l'extensibilité, la scalabilité des données…etc.

3 MIGRATION VERS LES SAAS

Les éditeurs de logiciels sont donc appelés à changer leur pensée pour passer de la fourniture des logiciels traditionnels « on-premise » à la fourniture des applications SaaS. Pour ce faire, ils doivent se focaliser sur trois domaines interdépendants (Chong & Carraro, 2006) qui sont :

- le modèle du business « business *model »,*
- l'architecture de l'application *« application architecture »* et
- la structure opérationnelle *« operational structure »*.



Dans les trois sections suivantes, nous examinerons de plus près chacun de ces changements, en se concentrant principalement sur l'aspect de l'application architecture SaaS.

3.1 Changer le Modèle du Business

Le changement du modèle du business peut impliquer un ou plusieurs des changements suivants :

- Changer la "possession" du logiciel : de la part du client à un fournisseur externe.
- Réaffectation des responsabilités de l'infrastructure technologique et de la gestion (matériels et services professionnels) : de la part du client au fournisseur.
- Réduction du coût de la fourniture des services de logiciel : par le biais de la spécialisation et l'économie tirée de la monté à échelle (la scalabilité).
- Cibler la « *long tail* ou longue traîne[11]» des plus petites entreprises, en réduisant le coût minimal pour lequel le logiciel peut être vendu (Anderson, 2006).

La réalisation des avantages de SaaS nécessite, à la fois, des changements dans les pensées des fournisseurs et ceux des clients. Cependant, c'est aux fournisseurs qu'appartient la tâche de convaincre les clients à effectuer ce changement.

3.1.1 Qui « possède» le logiciel ?

Avec le modèle de logiciel en tant que produit, qui domine le marché du logiciels, l'idée du logiciel en tant que service peut paraître un peu exotiques : au lieu "de posséder" complètement des logiciels importants, les clients peuvent payer pour s'abonner aux logiciels qui s'exécutent sur des serveurs distants. Dès qu'ils arrêtent l'abonnement, le logiciel disparait automatiquement. Il est donc particulièrement important que l'éventuel client comprenne comment le SaaS offre un avantage économique direct et quantifiable par rapport au modèle traditionnel.

3.1.2 Le Transfert de responsabilités informatiques

Dans une organisation typique, le budget consacré à l'IT est dépensé dans trois grands domaines :

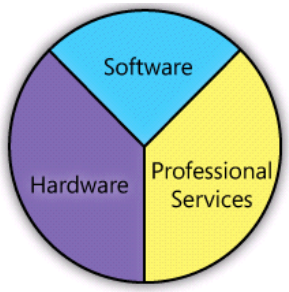
- **Les logiciels** : c'est l'ensemble des programmes et des données que l'entreprise utilise pour l'informatique et le traitement de l'information.
- **Le matériel** : c'est l'ensemble des ordinateurs de bureau, des serveurs, des composants réseau et des périphériques mobiles qui permettent aux utilisateurs d'accéder aux logiciels.
- **Les services professionnels** : c'est l'ensemble des personnes et institutions qui garantissent le bon fonctionnement et la disponibilité du système, y compris le staff du support technique, les consultants et les représentants du fournisseur.

Parmi ces trois domaines, le plus directement impliqué dans la gestion de l'information est le domaine des logiciels, qui représente le but ultime de toute organisation IT. Nous remarquons par exemple que toute organisation ajoutera des fonctionnalités logiciels sans matériel supplémentaire si elle pouvait le faire efficacement, mais aucune organisation n'ajoutera simplement du matériel sans un besoin prévu pour ajouter du logiciel (Chong & Carraro, 2006).

11 http://fr.wikipedia.org/wiki/Longue_tra%C3%AEne

3.1.2.1 Modèle d'un environnement IT basé sur les logiciels traditionnels

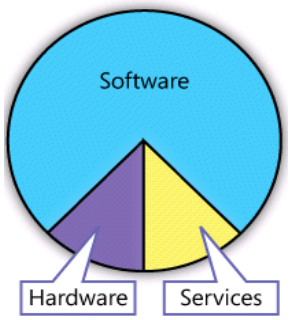
Comme c'est montré dans le diagramme suivant[12], dans un environnement IT basé sur les logiciels traditionnels, la majorité du budget est généralement dépensée sur le matériel et les services professionnels, laissant une minorité du budget disponible pour le logiciel (Chong & Carraro, 2006).



3.1.2.2 Modèle d'un environnement IT basé sur les SaaS

Dans une organisation qui s'appuie essentiellement sur le SaaS, la répartition du budget IT semble beaucoup différente, comme c'est illustré dans le diagramme ci-dessous.

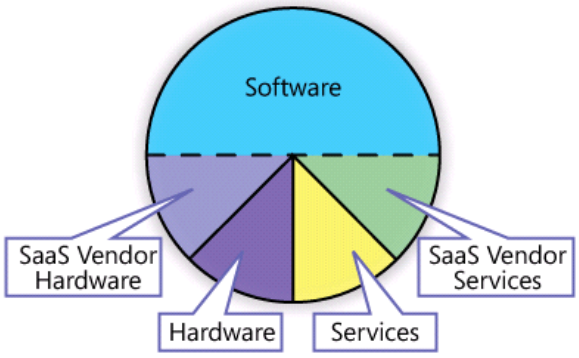
Dans ce modèle, le fournisseur de SaaS héberge des applications critiques et les données associées sur des serveurs distant, et il prend en charge la configuration matérielle et logiciel avec le staff de support dédié. Cela soulage l'organisation du client des responsabilités de la prise en charge du logiciel hébergé d'une part et de l'achat et l'entretien du matériel de serveur d'autre part (Chong & Carraro, 2006).



3.1.3 L'économie tirée de la scalabilité

L'économie réalisée par le paradigme SaaS est relié, dans sa grande partie, à la scalabilité. Par exemple, une application SaaS installée sur une ferme de cinq serveurs peut-être capable de prendre en charge 50 clients. Ce qui signifie que chaque client ne serait responsable que sur un dixième du coût d'un serveur. Une application similaire installée localement peut obliger chaque client de consacrer un serveur entier à savoir plusieurs (si l'équilibrage de la charge et la haute disponibilité sont exigés). Cela représente une économie substantielle et potentielle par rapport au modèle traditionnel.

De ce fait, les clients peuvent toujours obtenir beaucoup plus de fonctionnalités importantes de logiciels pour le même budget même après avoir calculer le matériel et les services professionnels acquis par les fournisseurs de SaaS comme c'est montré dans le diagramme suivant (Chong & Carraro, 2006).



12 Les proportions illustrées sur ces diagrammes sont uniquement pour faire des illustrations ; ils ne visent pas à défendre toute allocation spécifique de ressources.

3.2 Architecture d'application

Nous avons déjà présenté une définition du SaaS qui est "un logiciel déployé en tant que service hébergé et accessible sur Internet ». Cette définition peut englober beaucoup d'autres choses. Par conséquent, pour définir ce qu'on pourrait éventuellement appeler une application SaaS mature, nous devons présenter certains critères supplémentaires.

Dans ce qui suit, nous présentons en détail les différents attributs d'une architecture d'application SaaS.

3.2.1 Les attributs d'une architecture Multi-client & unique-Instance

D'un point de vue des architectes d'application, il y a trois atouts concurrentiels qui séparent une application SaaS bien conçue d'une autre mal conçue. Une application SaaS bien conçue est : scalable, efficacement multi-client et configurable.

- La scalabilité ou la monté à l'échelle d'une application : signifie la maximisation de la simultanéité et l'utilisation des ressources de l'application d'une manière plus efficace. Par exemple : l'optimisation de la durée de verrouillage, le partage des ressources communes telles que les threads et les connexions réseau, la mise en cache de données de référence et partitionnement des grandes bases de données (Chong & Carraro, 2006) (Jiehui, Ya, Jianqing, Jiyi, & Zhijie, 2010) (Liu G. , 2010).

- Le paradigme Multi-client : peut être considérer comme le plus important, surtout pour les personnes habituées à concevoir des applications isolées ou unique-client. Par exemple, quand un utilisateur d'une société accède à des informations d'un client à l'aide d'un service d'application CRM, il se connecte à une instance d'application qui peut être utilisée par d'autres utilisateurs depuis des dizaines ou des centaines d'autres sociétés (tout en gardant la transparence). Cela nécessite une architecture qui optimise le partage des ressources entre clients, et capable de distinguer les données appartenants à des clients différents (Chong & Carraro, 2006) (Jiehui, Ya, Jianqing, Jiyi, & Zhijie, 2010) (Pervez, Lee, & Lee, 2010) (Liu G. , 2010).

- Bien entendu, si une instance d'application unique sur un seul serveur doit accueillir à la fois des utilisateurs de plusieurs sociétés différentes, ce n'est pas possible d'écrire simplement un code spécifique pour personnaliser l'occurrence de chaque l'utilisateur final : la personnalisation du code d'application pour un client entraine un changement de celle-ci pour tout les autres clients. Au lieu de personnaliser l'application au sens traditionnel du terme, il faut donner à chaque client la possibilité d'utilise des métadonnées pour configurer la manière dont l'application apparaît et se comportée pour lui. Le défi relevé pour les architectes de SaaS est de veiller à ce que la tâche de configuration des applications soit simple et facile pour les clients, sans encourir un développement supplémentaire ou un coût d'opération de plus pour chaque configuration (Chong & Carraro, 2006) (Jiehui, Ya, Jianqing, Jiyi, & Zhijie, 2010) (Nitu, 2009).

3.2.1.1 Le modèle mature de SaaS

La première définition de SaaS est améliorée par l'ajout et l'identification des attributs importants d'une application SaaS mature. Mais le problème c'est que la maturité n'est pas une proposition «tout ou rien». Une application peut posséder seulement un ou deux des attributs déjà cités et peut répondre en même temps à tous les besoins nécessaires. Dans ce cas, les architectes d'application peuvent choisir de ne pas remplir les autres attributs si cela ne serait pas rentable.

D'une manière générale, la maturité des applications SaaS peut être exprimée à l'aide d'un modèle avec quatre niveaux distincts. Chaque niveau se distingue du précédent par l'ajout d'un des trois attributs répertorié ci-dessus. (Chong & Carraro, 2006) (Mietzner, Leymann, & Papazoglou, 2008).

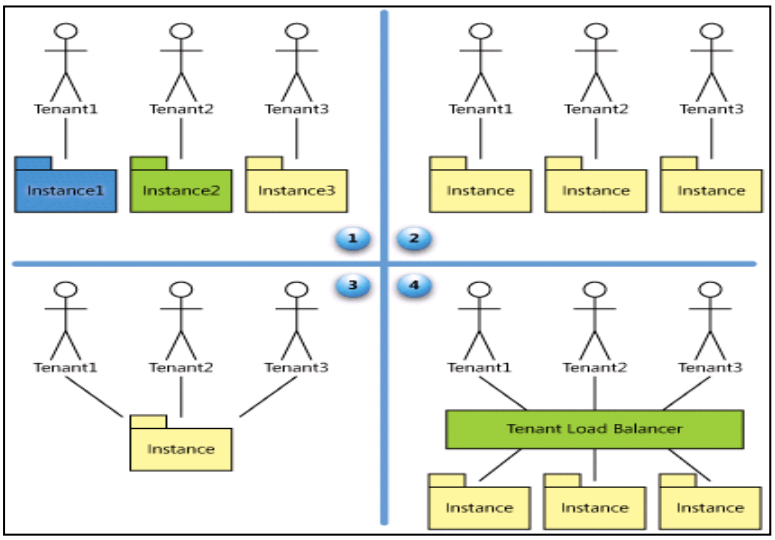


Figure 9. Les quatre niveaux du modèle de maturité Saas

Niveau I : Ad hoc/personnalisé : Le premier niveau de maturité est similaire au modèle traditionnel de livraison du logiciel « ASP ». A ce niveau, chaque client dispose sa propre version personnalisée de l'application hébergée et exécute sa propre instance de l'application sur des serveurs distants. Typiquement, les applications traditionnelles client/serveur peuvent être migré vers un modèle SaaS du premier niveau de maturité, avec un effort de développement relativement peu et sans réorganisation de l'ensemble du système. Cependant, ce niveau offre très peu des avantages d'une solution SaaS pleinement mature qui permet aux fournisseurs de réduire les coûts en consolidant le matériel et l'administration du serveur (Chong & Carraro, 2006) (Jiehui, Ya, Jianqing, Jiyi, & Zhijie, 2010).

Niveau II : Configurable : Au deuxième niveau de maturité, toutes les instances utilisent la même mise en œuvre du code. Le fournisseur répond donc aux besoins des clients en fournissant des options de configuration détaillées qui permettent à ces derniers de modifier l'apparition et le comportement de l'application en fonction de leurs exigences. Bien qu'il soit identique au niveau du code, chaque instance reste entièrement isolée de toutes les autres (Chong & Carraro, 2006) (Jiehui, Ya, Jianqing, Jiyi, & Zhijie, 2010). L'utilisation d'un code unique pour l'ensemble des clients réduit la possibilité de satisfaire les exigences en service d'une application SaaS, parce que les modifications apportées au code de base modifient l'application pour l'ensemble des clients à la fois.

Toutefois, le repositionnement d'une application classique en une application SaaS au deuxième niveau de maturité exige plus de réorganisation que le premier niveau, si elle a été conçue pour avoir des personnalisations individuelle plutôt que des métadonnées de configuration. Le second niveau de maturité, tout comme le premier niveau, nécessite la mise en œuvre de suffisamment de matériel et d'espace de stockage pour prendre en charge un nombre potentiellement important d'instances d'application s'exécutent simultanément (Chong & Carraro, 2006).

Niveau III : Configurable, Multi-client : Au troisième niveau de maturité, le fournisseur exécute une instance unique qui sert tous les clients, avec des métadonnées configurables. Les stratégies d'autorisation et de sécurité garantissent que les données de chaque client sont conservées

séparément des celles d'autres clients ; et que, du point de vue de l'utilisateur, il n'y a aucune indication que l'instance de l'application est partagée entre plusieurs clients. Cette approche élimine la nécessité de fournir l'espace mémoire (du côté serveur) pour autant d'instances que des clients. Cela permet une utilisation beaucoup plus efficace des ressources que le deuxième niveau, qui se traduit directement par la réduction des coûts. (Chong & Carraro, 2006).

Niveau IV : Scalable, configurable, Multi-client : Au quatrième niveau de maturité, le fournisseur héberge plusieurs clients sur une ferme d'équilibrage de charge des instances identiques, tout en conservant les données des clients séparées les unes des autres et en utilisant des métadonnées configurables.

Un système SaaS de ce niveau est arbitrairement scalable pour un grand nombre de clients. Le nombre de serveurs et d'instances peut être augmenté ou diminué pour répondre aux besoins, sans nécessiter une réorganisation supplémentaire de l'application. Les modifications ou les corrections peuvent être déployées pour des milliers de clients aussi facilement qu'un seul client (Chong & Carraro, 2006).

3.2.1.2 Le choix d'un niveau de maturité

Quel niveau de maturité doit être choisi pour une application ? On peut s'attendre à ce que le quatrième niveau est l'objectif ultime pour n'importe quelle application SaaS, mais cela n'est pas toujours le cas. Il peut être plus utile de penser à la maturité des SaaS en tant qu'un tout entre les données isolées et le code d'un côté et les données partagées et le code de l'autre côté (voir figure 10).

Figure 10. La maturité des SaaS en tant qu'un continuum

Lorsqu'on doit choisir un niveau de maturité d'une application au long de ce continuum, il faut prendre en considération que cela dépend des besoins opérationnels, architecturals, économiques et aussi les considérations du client. Toutes ces considérations sont interdépendantes les unes des autres dans une certaine mesure, comme c'est expliqué ci-dessous :

- **Le modèle économique** : une approche isolée, peut–elle avoir des gains financières ? le fait de négliger les avantages économiques et gestionnaires d'une approche partagée signifie une application offerte à un coût plus élevé. En outre, les clients peuvent avoir une forte résistance à un modèle architectural dans lequel plusieurs clients partagent l'accès à une application, même après avoir su que les données confidentielles sont bien protégées (Chong & Carraro, 2006) (Jiehui, Ya, Jianqing, Jiyi, & Zhijie, 2010).
- **Le modèle architectural** : est-ce qu'une application peut être faite pour être exécuté dans l'approche « unique-instance » ? Pour faire basculer une application traditionnelle on-permise ou client/serveur à une autre basée sur un système de livraison sur le net, il faut tenir en compte que :
  - l'application peut être incompatible avec une approche centrée sur les métadonnées et basée sur une instance unique, et
  - les gains financiers apportés par cette application ne peuvent même pas être suffisants pour l'investissement des efforts de développement nécessaire pour la transformer en une application SaaS pleinement mature.

Donc, l'élaboration et la conception d'une application basée sur le net, dès le début, donne probablement beaucoup plus de liberté pour adopter l'approche unique-instance (Chong & Carraro, 2006) (Jiehui, Ya, Jianqing, Jiyi, & Zhijie, 2010).

- **Le modèle opérationnel** : le contrat de niveau de service (SLA), peut-il être garanti sans isolation ? Il faut examiner soigneusement les obligations imposées par les SLAs avec les clients Déterminer aussi si ces obligations peuvent être respectées dans le cadre d'une architecture d'application multi-clients à instance unique (Chong & Carraro, 2006) (Jiehui, Ya, Jianqing, Jiyi, & Zhijie, 2010).

3.2.2 Haut niveau architectural

Architecturalement, les applications SaaS sont largement similaires aux autres applications créées à l'aide des principes de conception orientée service (Chong & Carraro, 2006), comme nous le montrons dans la figure ci-dessous.

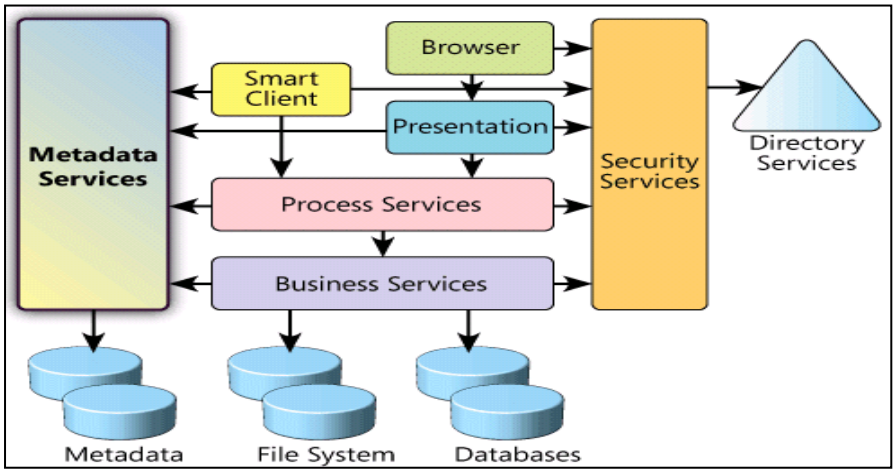


Figure 11. L'architecture des applications SaaS (Chong & Carraro, 2006) (Jiehui, Ya, Jianqing, Jiyi, & Zhijie, 2010)

La plupart des composants illustrés par cette figure devrait être familier, le plus, aux architectes d'application. Parmi les composants de cette architecture, on trouve :

- **Le processus services** qui expose :
  - Les interfaces que les *smart clients* et/ou la *Web présentation* peuvent invoquer, et
  - Les longues transactions qui invoquent d'autres *business services*, et qui interagissent avec les données correspondantes stockées pour lire et écrire des données de business « *business data».*

- **Les Services de sécurité** « *Security services* » qui sont responsables sur le contrôle d'accès aux services de logiciels.
- Et **les services de métadonnées** «*metadata services*», qui sont responsables sur la gestion et la configuration de l'application pour les clients. L'ensemble des services et les *smart clients* interagissent avec les services de métadonnées afin de récupérer des informations qui décrivent les configurations et les extensions qui sont spécifiques à chaque client.

Les **services de métadonnées** sont considérés comme la différence la plus importante entre les applications SaaS et tous les autres type d'applications. Nous allons détailler un peu plus les

services de métadonnées et les services de sécurité dans ce qui suit vu leur importance pour les applications SaaS.

3.2.2.1 Les services Métadonnées

Dans une application SaaS mature, le service de métadonnées offre aux clients le principal moyen de personnalisation et de configuration de l'application pour répondre à leurs besoins. Généralement, les clients peuvent modifier la configuration dans quatre grands domaines qui sont :

1. **L'interface utilisateur** : les applications SaaS offrent généralement des fonctionnalités qui permettent aux clients de changer l'interface
2. **Les Règles de flux de travail et du business** : afin d'être pratique pour une large gamme de clients potentiels, une application SaaS critique doit pouvoir tenir compte des différents flux de travail « *workflow* ».
3. **Modèle de données extensible** : Un modèle de données extensible offre la possibilité aux clients de personnaliser l'application pour qu'ils puissent la faire fonctionner à leur façon. Plus loin dans ce document, nous présentons comment un modèle de données extensible est conçu.
4. **Contrôle d'accès** : en général, chaque client est responsable de la création des comptes individuels et de la détermination des ressources et des fonctions que chaque utilisateur est autorisé à y accéder.

Pour fournir aux clients la flexibilité pour la configuration d'un logiciel au tant que nécessaire, les options sont organisées en unités de configuration hiérarchique appelées « *scopes».* Chacune de ces dernières contient des options permettant d'apporter des modifications dans chacun des quatre domaines énumérés ci-dessus. (Chong & Carraro, 2006) (Jiehui, Ya, Jianqing, Jiyi, & Zhijie, 2010).

Contrairement aux applications de la ligne-de-business traditionnelles, les applications SaaS sont beaucoup plus susceptibles d'être configurés par les clients eux-mêmes. La conception de l'interface de configuration est donc presque aussi important que la conception de l'interface pour les utilisateurs finaux.

3.2.2.2 Services Sécurité

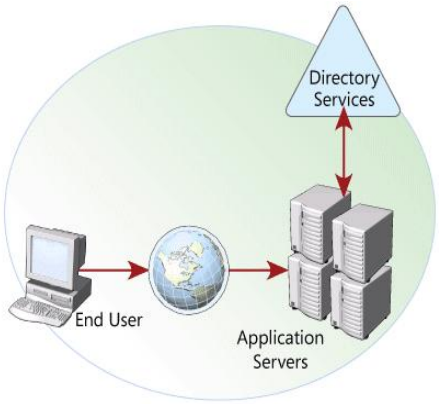
D'une manière générale, la sécurité est importante pour n'importe quel logiciel en particulier les SaaS. Leur nature rend la sécurité une préoccupation primordiale pour les clients et une priorité élevée pour les architectes d'application.

De plus, le suivi de quelques directives de base peut aider à garantir que les clients gardent le contrôle sur leurs données privées. Parmi ces directives, nous citons :

L'authentification : les concepteurs de SaaS utilisent deux approches générales de gestion de l'authentification : un système d'authentification centralisé, ou un système d'authentification décentralisé (Chong & Carraro, 2006).

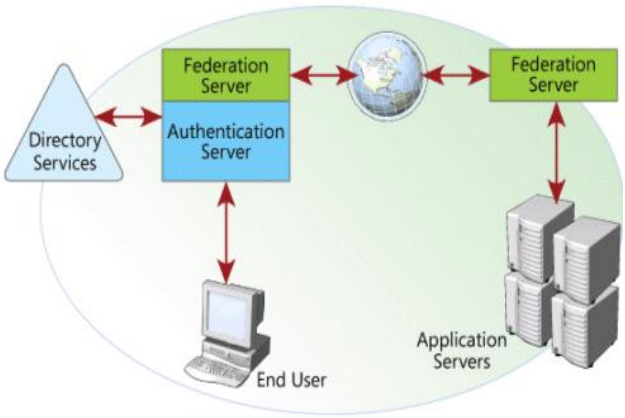
Dans **un système d'authentification centralisé**e, le fournisseur gère une base de données de compte utilisateur central qui sert l'ensemble des clients de l'application. L'administrateur du client est autorisé à créer, gérer et supprimer des comptes d'utilisateur du répertoire des comptes utilisateurs. Un utilisateur accède à l'application en fournissant son identifiant (Chong & Carraro, 2006).

L'application à son tour authentifie les informations d'identification dans le répertoire central et accorde l'accès à l'utilisateur si les informations d'identification sont valides.



Cette approche nécessite une infrastructure d'authentification relativement simple, facile à concevoir et à mettre en œuvre.

Dans un **système d'authentification décentralisé**, le client déploie un service de fédération qui interface avec son propre répertoire de service « directory service». Lorsqu'un utilisateur tente d'accéder à l'application, le service de fédération authentifie l'utilisateur localement et émet un jeton de sécurité. Le système d'authentification de fournisseur SaaS accepte le jeton et permet à l'utilisateur d'accéder à l'application (Chong & Carraro, 2006).



Cependant, l'approche décentralisée est plus complexe que l'approche centralisée.

L'autorisation : en général, l'accès aux ressources et aux fonctions de l'entreprise dans une application SaaS est géré à l'aide de rôles correspondants aux fonctions spécifiques au sein d'une organisation. A Chaque rôle, on a attribué une ou plusieurs autorisations permettant aux utilisateurs affectés au rôle d'effectuer des actions en conformité avec les règles pertinentes de l'entreprise (voir la figure ci-dessous) (Chong & Carraro, 2006).

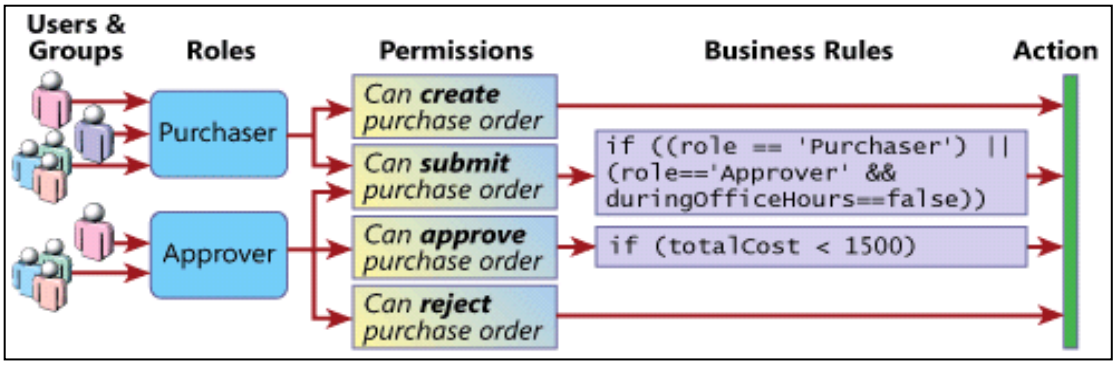


Figure 12. Contrôle d'accès (Chong & Carraro, 2006)

Les rôles sont gérés au sein de l'application SaaS elle-même :

- Ils peuvent contenir des comptes d'utilisateurs individuels, ainsi que des groupes d'utilisateurs.
- Les comptes d'utilisateurs individuels et les groupes peuvent être affectés à plusieurs rôles différents selon les besoins.
- Selon les rôles auxquels un utilisateur est affectés, on lui accord une ou plusieurs autorisations pour effectuer des opérations spécifiques ou des actions.
- Généralement, ces actions sont directement liées aux importantes fonctions de l'entreprise, ou à la gestion de l'application elle-même.

3.2.3 La Scalabilité

Les logiciels d'entreprise largement scalable sont destinés à être utilisés par des milliers de personnes simultanément. La création d'applications d'entreprise de ce genre mène à découvrir les défis de la création d'une architecture scalable (Chong & Carraro, 2006) (Chong, Carraro, & Wolt, 2006).

Pour une application SaaS, la scalabilité est le critère le plus important. Il faut prendre en charge en moyenne la base utilisateur d'un seul client, multiplié par le nombre total de clients de l'application. La scalabilité peut être étudiée aux niveaux des applications et des bases de données.

A. La monté à l'échelle de l'application

Les applications peuvent être *scaled up* (en déplaçant l'application sur un serveur plus grande et plus puissant) ou *scaled out* (en exécutant l'application sur plusieurs serveurs). Cependant, au niveau des SaaS, l'exécution de l'application sur plusieurs serveurs (scaling out) est presque toujours la meilleure façon d'ajouter de la capacité, comme nous l'avons représenté plus haut, dans le modèle de maturité de SaaS. Une application SaaS bien conçue peut être exécutée sur arbitrairement un grand nombre de serveurs, chacun d'eux exécute une ou plusieurs instances identiques. Voici quelques conseils pour la conception d'une application qui peut être exécutée sur plusieurs serveurs «scale out»[13]:

B. La monté à l'échelle des données

Du fait que les bases de données servent un plus grand nombre d'utilisateurs simultanément, la quantité de temps nécessaire pour effectuer des opérations telles que l'interrogation et la recherche augmente considérablement. Les applications SaaS, qui utilisent souvent les mêmes bases de données pour servir des milliers de clients, sont particulièrement sensibles à ces types de dégradation des performances. Il est donc important de planifier la croissance adéquatement (Chong & Carraro, 2006) (Chong, Carraro, & Wolt, 2006).

Afin d'améliorer l'efficacité des requêtes et des mises à jour, une façon assez simple pour la monté à l'échelle d'une base de données est proposée. Il s'agit du partitionnement et de la division des données en petits segment « *chunks* ». Sans oublier la nécessitée d'élaborer une stratégie de partitionnement pour déterminer la meilleure façon de partitionnement des données (Chong & Carraro, 2006) (Chong, Carraro, & Wolt, 2006). Par exemple, si une application a des clients de partout dans le monde, une stratégie de partitionnement géographique pourrait être appropriée.

13 http://msdn.microsoft.com/practices/Topics/perfscale/default.aspx

3.3 Structure opérationnelle

Le troisième changement important dans la pensée a une relation avec la structure opérationnelle de l'application. Autrement dit, ce qu'il faut faire pour assurer l'application et la maintenir bien disponible. Les fournisseurs de SaaS doivent non seulement être des experts dans la création du logiciel et sa distribution sur le marché, mais également des experts dans le fonctionnement et la gestion de ce dernier. Pour de nombreux éditeurs de logiciels traditionnels, c'est peut-être l'aspect le plus inhabituel.

Nous nous sommes déjà familiarisés avec les principes fondamentaux de l'hébergement et de services Web à travers l'expérience avec les sites des entreprises présentées sur l'échelle d'internet. Cependant, le service d'hébergement est responsable sur la disponibilité du site, mais il n'est pas généralement responsable sur l'activation et l'entretien du site (Chong & Carraro, 2006). La fourniture des logiciels en tant que service ajoute une couche supplémentaire à prendre en considération lors de la création des arrangements hébergés (voir figure ci-dessous).

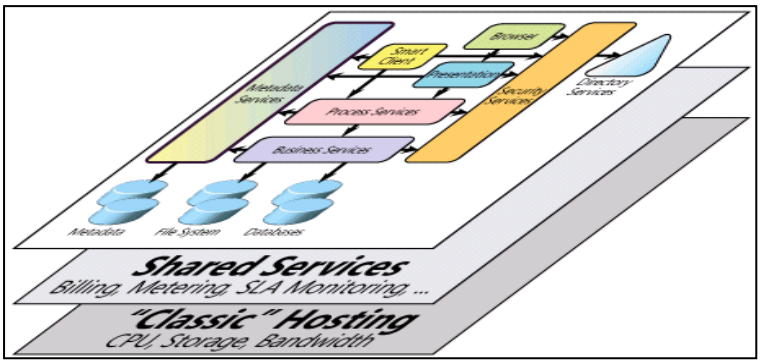


Figure 13. Couche supplémentaire pour l'ébergement des SaaS (Chong & Carraro, 2006)

Le principe de l'hébergement des sites a développé le besoin d'avoir un système de paramétrage et de facturation afin d'effectuer les opérations suivantes :

- Assurer le suivi de l'utilisation et la facturation du service en fonction de temps ou des ressources utilisés par les clients.
- Limiter l'accès pendant certains moments de la journée, afin de satisfaire d'autres critères.
- Surveiller l'accès au site et aux performances, pour s'assurer que les niveaux de service SLAs sont respectés.
- Exécuter d'autres fonctions afin d'assurer une expérience transparente pour les clients qui répondent ou dépassent les attentes.

Collectivement, les systèmes utilisés pour effectuer ces fonctions sont appelées des services partagés. Tel que l'hébergement traditionnel du web, il faut créer la couche de services partagés puis héberger l'application pour la rendre disponible. Les Fournisseurs de SaaS proposent un ensemble de services partagés pour contrôler le business et les problèmes opérationnels identifiés ci-dessus.

Parmi ces services, il y en a principalement ceux qui sont consacré à la surveillance pour assurer la disponibilité et d'autre pour surveiller le niveau des performances.

1. **La surveillance afin d'assurer la disponibilité[14] :** La haute disponibilité doit être l'une des plus importantes priorités pour n'importe quel fournisseur de SaaS. Une panne qui affecte un seul serveur ou un centre de données pourrait conduire à des pertes importantes des données ou à la diminution de la productivité pour un grand pourcentage des clients. La prise en charge des techniques de base, telles que la surveillance de la pulsation et des mécanismes d'alerte dans l'application est recommandé. Il faut aussi assurer qu'il existe des processus dans le centre d'opérations pour continuer à fonctionner même en cas de panne du système (tolérance aux fautes).
2. **La surveillance des performances** : Les clients attendent à ce qu'on leur offre un accès aux applications à un niveau de performance acceptable. A un certain niveau, cette expectative est explicite par les termes du contrat de niveau de service qui est engagé sur l'honneur avec le client. Cependant, Au-delà des SLAs, si les clients perçoivent que l'application est lente ou ne répond pas, ils seront plus susceptibles de mettre fin ou de refuser de renouveler leurs abonnements. Les utilisateurs mécontents peuvent faire circuler leurs sentiments sur les sites web, donnant ainsi à l'application une réputation négative.

Afin de garantir un haut niveau de performance, il faut :

1. Construire un support pour des compteurs de performance dans l'application directement si c'est possible.
2. Définir des seuils de performances pour les différents paramètres telles que le temps de réponse de l'application et le taux d'utilisation de CPU, et
3. Utiliser des alertes pour avertir le personnel approprié quand des événements de la gestion sont augmentés.

4 CONCLUSION

Le SaaS représente un nouveau paradigme dans la livraison du logiciel, un modèle architectural construit sur les principes d'efficacité, de clients multiples, de scalabilité et configurabilité pilotés par les métadonnées pour fournir un bon logiciel avec peu de frais pour les clients existants et potentiels. Cependant beaucoup de confusion se pose quand il s'agit de la distinction entre les SaaS d'une part et les autres technologies et architectures d'une autre part tels que le SAP et le SOA. Dans le chapitre suivant, nous faisons une comparaison entre le SaaS et ces derniers afin de les distinguer.

14 http://www.microsoft.com/technet/itsolutions/cits/mo/smf/smfavamg.mspx

TROISIEME CHAPITRE : ETUDE COMPARATIVE ENTRE SAAS, ASP ET SOA

1 INTRODUCTION

La différence entre les SaaS et les autres technologies et architectures n'est pas bien définie. Par exemple, entre SaaS et ASP : l'ASP traditionnel se limite à la fourniture d'applications en mode hébergé tandis que le SaaS désigne une application modulaire comprenant des outils et des personnalisations pour répondre aux besoins des clients. SaaS et SOA sont de nouvelles technologies importantes adoptées de plus en plus par les entreprises. Néanmoins, les deux sont parfois mal compris. En définissant bien ses deux termes et en précisant les différences entre eux, on peut aider les concepteurs et les développeurs dans le choix d'architecture et du modèle approprié.

Dans ce chapitre, on va voir la nécessité de bien définir tous les termes qui ont une relation avec les SaaS. On a fait des comparaisons entre les SaaS et les ASP d'une part et entre les SaaS et SOA d'une autre part. En dernier lieux, on a cité les différents bénéfices et les freins des SaaS.

2 SAAS VS ASP (APPLICATION SERVICES PROVIDERS)

SaaS et ASP font partie tous les deux de la catégorie des logiciels métiers.

- L'ASP adopte une architecture à une tendance unique ; où chaque application cliente est conçu à la demande et elle est mono utilisateur. Donc, elle ne peut pas prendre en considération un autre utilisateur. La technologie ASP utilise la même base de données et un catalogue virtuel. Les clients qui ont utilisé des applications ASP doivent payer les frais d'utilisation et la mise à jour du logiciel ainsi que les frais de gestion du serveur.
- Alors que SaaS adopte une architecture Multi-tenancy ; où chaque application est offerte aux utilisateurs multiples. Le mode SaaS n'est pas seulement limité aux applications administratives, mais il fournit des services de soutien professionnel à travers Internet. Il met aussi l'accent sur la personnalisation des applications logicielles.

En général, un logiciel SaaS est similaire à un logiciel à la demande, un logiciel ASP et un logiciel hébergé. Ils sont tous déployés sur les serveurs distants des fournisseurs d'applications et loués par le biais d'Internet en fonction des besoins réels des utilisateurs. Ceux-ci paient les frais en fonction du nombre de fonctions du logiciel louées ou de la durée du temps d'utilisation (Liao & Tao, 2008). D'après la définition faite par AMR Research[15] à Boston, SaaS comprend les fonctionnalités suivantes:

- Les fournisseurs de logiciels n'ont pas à demander une énorme somme de frais pour la vente des licences des logiciels, mais seulement les frais d'utilisation de ces derniers (Liao & Tao, 2008).
- Les clients n'ont pas à investir en d'autres matériels informatiques, à l'exception des PC et des logiciel de connexion à Internet (Liao & Tao, 2008).
- Les concepts « Multi-tenancy » et « unique-instance » sont adopté (Liao & Tao, 2008).

Les solutions ASP sont historiquement des solutions de type client / serveur dans lesquelles le prestataire héberge le serveur. La solution SaaS s'appuie sur les standards web pour proposer des applications accessibles par des clients légers. On retiendra donc que l'ASP et le SaaS sont identiques dans leurs intentions (vendre un service plutôt qu'un logiciel à installer), mais différentes dans leur réalisation technique : seul le SaaS reproduit des architectures web (SOAP, XML, etc.). Dans le tableau 2, nous essayons de résumer toutes les différences entre les SaaS et les ASP.

15 http://en.wikipedia.org/wiki/AMR_Research

	Application Service Provider (ASP)	Software as a Service (SaaS)
Le déploiement des applications	Emprunté. L'ASP déploie des applications commerciales des autres entreprises. C'est la raison pour laquelle l'avantage du coût est faible et les capacités de personnalisation sont limitées.	Construit. Le logiciel est développé par le fournisseur SaaS dès le début et l'avantage du coût est maximum.
Temps d'implémentation	Long. L'installation et à personnalisation d'une application commerciale construite par une autre société prendra un cycle plus long.	Immédiate. Disponible pour tous les clients, à la demande et dans des délais précis.
Usabilité	Difficile. Une version personnalisée d'un système déjà complexe exige beaucoup de formation et d'orientation	Facile. Les applications SaaS basées sur Internet sont connues pour leur facilité d'utilisation intuitive, les utilisateurs peuvent démarrer le système immédiatement.
Le design d'applications	Des programmes client-serveur monolithiques. Les applications ASP étaient des programmes client-serveur monolithiques avec une interface web HTML simplifié.	Des programmes basés sur le web, facile à utiliser. Les solutions SaaS sont modernes, elles sont conçues pour l'environnement web, ce qui améliore la convivialité et simplicité de gestion
Mises à niveau et améliorations	Peu fréquents. Puisque l'ASP dépend souvent des fournisseurs de logiciels, leur capacité de mise à niveau de l'application est limitée. Les mises à jour sont déployés à chaque fois que le fournisseur d'application les émis, habituellement une fois par an ou moins. Le manque de multi-tenancy fait que les mises à niveau de l'instance unique est impossible.	Souvent. Les meilleurs pratiques et les meilleures idées sont incorporées comme des améliorations. Puisque aucun logiciel n'est déployé sur le site du client, les améliorations peuvent être mises en œuvre dans le Centre de données de SaaS, ensuite, elles sont mises à la disposition de l'ensemble de la communauté des utilisateurs. Le paramètre de configuration permet aux clients d'adopter ou de rejeter ces changements selon leurs besoins.
Intégration	Cher et chronophage (gourmande en temps)	Pas chère
IT support	Exclusive. Surveillance interne, selon le degré de personnalisation et d'intégration, des besoins de maintenance sont ajoutés.	Inclusive. Une partie du service est incluse.
Multi-tenant Scalability	Non. Chaque client est maintenu dans un environnement ASP ; aucun moyen pour monter à l'échelle à travers les silos.	Oui. Les applications sont conçus pour être utilisé dans un environnement multi-locataires, les systèmes sont configurés pour que chaque client peut avoir une version personnalisée.
L'acceptation du marché	Moin. Les fournisseurs ASP bâclé leurs offres avant que les problèmes de performances, de sécurité, de personnalisation et d'intégration ne soient résolus, et avant que l'IT de l'organisation cliente ne soit prête à adopter la solution ASP.	Plus. Aujourd'hui, les entreprises et les utilisateurs sont mieux équipés pour tirer parti des SaaS comme l'expérience de l'informatique est amélioré avec les technologies web et l'approche orienté-services. Les besoins du business tels que la conformité législation prouve que c'est le moment pour que les entreprises adoptent une approche SaaS.
La conformité du matériel	Difficile. Une page ASP prendrait un logiciel qui n'a pas était écrit pour être hébergés et elle le place sur le web. Cela peut souvent être la cause des problèmes de conformité de matériels.	Facile. La plupart des produits SaaS sont conçu spécialement pour une utilisation sur l'internet. Aussi,le matériel d'hébergement des SaaS est développé pour une utilisation sur internet.
Market timing	En avance sur son temps. Le modèle ASP est un peu en avance ahead sur son temps quand la vitesses à Internet très faibles.	À temps. Aujourd'hui, à cause de l'augmentation de la vitesse de la connexion internet, il est possible de transférer les données rapidement et facilement sur internet.

Tableau 2. Les principales différences entre SaaS et ASP (luitinfotech)

3 SAAS VS SOA

Même les professionnels de l'informatique, à un moment ou un autre, à tort interchangent les termes SaaS et SOA. Au mieux, cette pratique défectueuse engendre de la confusion ; au pire, il peut mener à la mauvaise conception. Notre objectif est donc de clarifier le sens de ces deux termes souvent utilisés et maltraités.

En bref, la différence entre SaaS et SOA est que le premier est un modèle de livraison de logiciel alors que le deuxième est un modèle de construction de logiciel. Dans ce qui suit, nous examinons brièvement les concepts de SaaS et SOA. Nous décrivons les différences entre SaaS et SOA du point de vue architecture du logiciel.

3.1 Définitions

Parfois appelé logiciel d'abonnement (Turner, Budgen, & Brereton, 2003), le modèle de livraison SaaS sépare essentiellement la propriété du logiciel de l'utilisateur. Le propriétaire est un vendeur qui héberge le logiciel et qui permet à l'utilisateur de l'exécuter à la demande via internet ou un intranet.

Dans un modèle SOA, les éléments constitutifs du système logiciel sont réutilisables (Zhang, Zhang, & Cai, 2007). La collection de services interagissent entre eux grâce à des interfaces standards et des protocoles de communication. SOA promet de changer fondamentalement la façon dont nous construisons des systèmes internes ainsi que la façon dont les systèmes internes et externes interagissent.

Malgré leurs significatives différences, SaaS et SOA sont des modèles d'architecture strictement liés pour les systèmes d'information à grande échelle. À l'aide de SaaS, un vendeur peut livrer un système logiciel en tant que service. L'utilisation du SOA permet aux services d'être publiés, découverts et adoptés comme un composant de service pour construire un nouveau logiciel, qui peut être aussi publié et mis en œuvre en tant qu'un nouveau service. En d'autres termes, les deux modèles se complètent : SaaS permet d'offrir des composants pour SOA à utiliser, et SOA permet de réaliser rapidement des SaaS. (Laplante, Zhang, & Voas, 2008).

Bien que les deux offrent des fonctionnalités prometteuses pour l'industrie du logiciel, ils représentent qu'un niveau conceptuel du modèle et ils exigent une technologie d'appuie. À l'heure actuelle, la meilleure technologie utilisée pour les SaaS et le SOA est la technologie de services web[16] (Laplante, Zhang, & Voas, 2008). En d'autres termes, les technologies de services web, avec leur pile de standards, permettent et facilitent l'utilisation de SaaS et de SOA. Il est à noter que ni SaaS ni SOA ne nécessitent une technologie de services web, mais c'est de loin la meilleure option actuelle pour les soutenir.

3.2 Architectures du logiciel

SaaS et SOA se concentreront sur les connexions entre les composants constitutifs. Par conséquent, ils appartiennent à la dimension du réseau (Laplante, Zhang, & Voas, 2008). Le tableau ci-dessous met l'accent sur les différences entre la SOA et SaaS .

16 Ferris, C., & Farrell, J. (2003). What Are Web Services?

Perspective	SOA	SaaS
Objectifs/Portée	Liste de services possibles à utiliser.	Liste de services possibles à livrer
Modèle du Business	Liste de services du business à utiliser	Liste de services du business à fournir
Modèle de système d'information	Modèle d'interaction des composants de service	Modèle d'interaction des composants
Modèle de la technologie	Dépendant de la technologie et la plate-forme selon le modèle d'interaction des composants de service	Dépendant de la technologie et la plate-forme selon le modèle d'interaction des composants
Représentation détaillée	La liste des langues dépendants de la technologie et les protocoles utilisés (tels que UDDI, SOAP, XML, WSDL) et des services utilisés actuellement.	Architecture basée sur la Publication/abonnement et la facilité de notification, la liste des langues dépendent de la technologie, les protocoles et les services utilisés.
Fonctionnement du système	La communication interservices, la coordination et la collaboration	La communication inter composant, la coordination et la collaboration.

Tableau 3. les différences entre la SOA et SaaS

Du point de vue objectifs/portée, le modèle SOA est une liste de services susceptibles d'être utilisées dans un logiciel en cours de construction ; tandis que pour le modèle SaaS, c'est une liste de services possibles d'être livrés.

Du point de vue du propriétaire, SOA contient la liste de services métiers trouvés pour être utilisés dans le système alors que SaaS contient la liste de services métiers qui doivent être fournis. En utilisant les services métiers existants, des frais significatifs de conception et de développement de logiciels peuvent être éliminés. Notez que SaaS ne signifie pas qu'un logiciel doit être est livré sous un seul service. Donc, cela signifie qu'un logiciel peut être livré sous de multiples services ; c'est-à-dire les parties du logiciel peuvent être des services autonomes qui travaillent avec le service maître pour l'ensemble du système.

Du point de vue du concepteur, SOA représente un modèle architectural décrivant les patrons des interactions entre les composantes des services, tandis que le SaaS décrit les patrons des interactions entre les éléments qui ne sont pas nécessairement des services.

Du point de vue du constructeur, SOA et SaaS doivent identifier une technologie (telle que les services web) pour réaliser les modèles d'interaction. La liste des langues détaillées et protocoles doivent également être identifiés — par exemple, WSDL pour la description, UDDI pour la publication et SOAP pour la communication.

En gardant SOA à l'esprit tout en créant un SaaS, les développeurs peuvent produire intentionnellement plusieurs services à divers niveaux granulaires. De cette façon, plus de services à divers niveaux de complexité peuvent devenir disponibles et ainsi faciliter davantage la construction des services basés sur la SOA.

Dans ce qui suit, nous présentons les avantages ainsi que les inconvénients des SaaS.

4 LES AVANTAGES DU SAAS

Même si les stratégies d'entreprise sont toutes différentes, on peut retenir quelques tendances fortes dans les stratégies informatiques à l'heure actuelle qui sous tendent les orientations vers le SaaS et le Cloud Computing :

- Réductions des coûts informatiques souvent considérés comme trop élevé par rapport au service rendu (Cao & Zhou, 2009).
- Concentration vers les activités métier différentes par rapport aux concurrents.
- Amélioration de la réactivité (« agilité ») aux changements de l'environnement et aux adaptations de la stratégie (y compris les modifications de périmètre de l'entreprise via les cessions, acquisitions, fusions)
- Innovation pour intégrer les évolutions technologiques dans les offres de produits et services de l'entreprise et dans ses processus (internes et avec ses partenaires).

Sur le modèle élaboré par Norton et Kaplan au début des années 90, le Cabinet Forrester a décliné une version adaptée aux perspectives des systèmes d'information (Symons, 2004). Le balanced scorecard est un outil permettant de mesurer les performances d'une organisation par rapport à une stratégie donnée. Il s'articule selon quatre dimensions (Symons, 2004) (Boucher, 2009) :

- Point de vue utilisateur : comment l'informatique doit apparaître aux yeux des utilisateurs ? (utilisateur s'entend ici au départ comme interne à l'entreprise mais de plus en plus inclus les clients et partenaires amenés à utiliser certaines parties du SI de l'entreprise).
- Point de vue de la contribution métier : comment l'informatique apparraît aux yeux de la direction générale comme un contributeur important au succès de l'entreprise ?
- Point de vue de l'excellence opérationnelle : quels sont les services et processus dans lesquels l'informatique doit exceller pour satisfaire les utilisateurs ?
- Point de vue de la préparation de l'avenir : comment l'informatique développe la capacité à changer et s'améliorer pour mieux atteindre ses objectifs stratégiques et ceux de l'entreprise ?

On peut lister des contributions du modèle SaaS selon les quatre dimensions précédentes :

Contribution métier	• 'Scalability' : le SaaS permet une adaptation rapide au périmètre de l'entreprise (à la hausse comme à la baisse) • Mise à jour rapide des fonctionalités • Dégagement de moyens humains pour l'innovation métier (en externalisant en SaaS les processus courants) • Capacités financières en convertissant des budgets d'investissement en coûts récurrents • Réduction des coûts et variabilisation (paiement à l'usage) • Facilité à identifier les coûts, les refacturer et valider leur pertinence
Utilisateurs	• Meilleure interface, facilité d'utilisation (influence de l'origine grand public du modèle, technologie RIA) • Facilité d'intégration avec les autres outils (mash up) • Evolutivité rapide : intégration plus rapide des retours utilisateurs
Excellence opérationnelle	• Industrialisation des opérations par les opérateurs SaaS • Niveaux de services (SLA) sont contractualisés en externe : plus facile à contractualiser en interne. Outillage de suivi des performances fournis souvent par le vendeur de SaaS. • Sécurité des données : grands opérateurs SaaS ont des certifications difficiles à obtenir en interne. • Déploiement simplifié : réduction des délais de mise en oeuvre des projets.
Préparation du futur	• Capacité à tester plus facilement de nouvelles applications et nouvelles technologies. (mode try and buy, plateforme de développement PaaS) • Évolution des ressources internes vers les besoins spécifiques métier • Nécessité de gérer des relations externes plus complexes : formation, mise en place de gouvernance spécifique (basée sur E-Scm?)

Tableau 4. les contributions du modèle SaaS

4.1 Bénéfices économiques

La réduction des coûts procurés par le modèle SaaS est le moteur premier de l'intérêt pour ce mode de livraison de logiciels. Ceci est principalement dû au facteur d'échelle entre une application gérée par l'entreprise et hébergée sur sa propre infrastructure et la même application hébergée pour de multiples clients par un prestataire spécialisé sur son infrastructure (Boucher, 2009) (Choudhary, 2007).

Ainsi, nous présentons quelques bénéfices économiques :

- réduction du coût total de possession : exemple de la messagerie : on s'appuiera sur une étude de Forrester centrée sur les services d'emails (Schadler, 2009), on trouve que pour les messageries intégralement supportées en interne, le coût mensuel par employé est de 25$/mois. Par comparaison, un modèle pur SaaS comme celui de Google Apps est évalué par Forrester à 8,50$/mois soit 3 fois moins. (Boucher, 2009).

- Réduction de coûts : exemple d'un déploiement de CRM : Dans une étude de 2007, le cabinet Mc Kinsey estime à 30% la réduction du TCO pour le déploiement d'un logiciel de CRM pour 200 utilisateurs (Dubey & Wagle, 2007) (Boucher, 2009). Les gains principaux portent sur :

    1. Le déploiement :

    a) la réduction du temps de déploiement (pas de déploiement sur le poste client mais aussi moindres possibilités de customisation, automatisation).
    b) l'absence de déploiement d'infrastructure.
    c) la formation est plus aisée : interface plus simple, autoformation proposée par le vendeur.

    2. Les opérations :

    a) La gestion des versions (y compris la prise en compte des retours utilisateurs) est prise en charge par le vendeur.
    b) L'infrastructure est gérée par le vendeur.
    c) La fiabilité (niveau de service) : en SaaS on obtient 99,9% contre en moyenne 99% en interne.
    d) l'adaptation à l'utilisation : pas de licence non utilisée.

- Moindre prise de risque au départ : En mode SaaS, il est facile de réaliser une phase pilote sans aucun engagement et sans rien dépenser en infrastructure (le mode « Try and Buy »). De plus, le paiement correspond à l'utilisation et le contrat peut être arrêté si la solution ne s'avère pas satisfaisante : la prise de risque est faible.

4.2 Bénéfices fonctionnels

Parmi les bénéfices fonctionnels du mode SaaS, nous trouvons principalement :

1. **La standardisation des processus :** Un des avantages du recours à des services hébergés pré-packagés est de faire entrer l'entreprise dans un mode de standardisation de ses processus (Boucher, 2009).

2. **L'apport de fonctionnalités à tout type d'entreprises :** La facturation à l'usage (au nombre d'utilisateurs par exemple) permet à de petites et moyennes entreprises d'accéder à des fonctionnalités réservées à de très gros clients (Boucher, 2009).

3. **L'amélioration de la disponibilité :** Le fonctionnement en mode SaaS permet généralement d'obtenir des taux de disponibilité plus importants qu'avec un logiciel hébergé. La disponibilité doit toutefois être évalué de « bout en bout » et tenir compte des disponibilités du réseau d'accès. Un autre aspect de la disponibilité est la possibilité de se connecter depuis d'autres terminaux que le poste de travail (Lu & Sun, 2009) (Boucher, 2009).

4. **L'apport de nouvelles formes de relation de travail dans l'entreprise** : Le passage d'une solution bureautique classique à Google Apps apporte en plus des applications traitement de textes, tableurs ou présentation toute une suite d'outils collaboratifs. Il est ainsi possible de travailler à plusieurs et simultanément sur le même tableur, chacun pouvant le modifier en temps réel.

   C'est ce qu'illustre le schéma suivant issu d'une étude McKinsey sur l'impact des nouveaux outils technologiques (ici nommés web 2.0) sur la productivité (Chui & al, 2009).

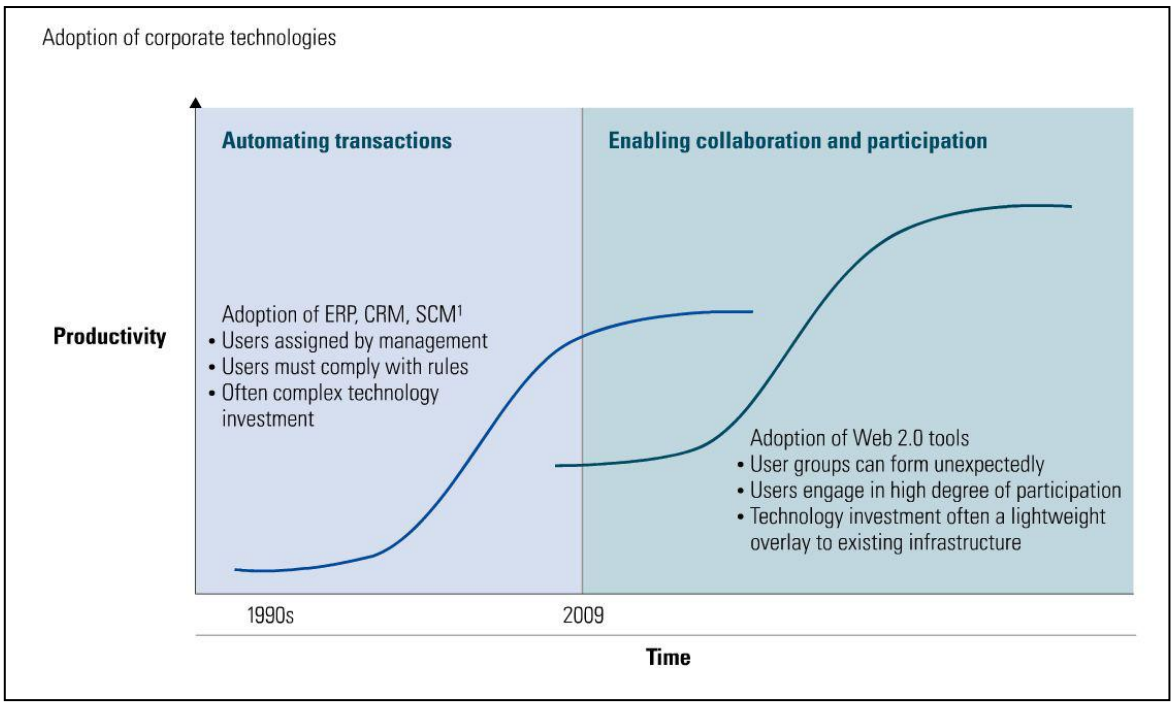


Figure 14. Six façons pour l'adoption des outils du Web 2.0 (Chui & al, 2009)

Après la phase d'automatisation des années 90 et 2000 via les gros progiciels intégrés (ERP) qui a permis une hausse de la productivité au prix d'investissements lourds, on entre dans une nouvelle phase au cours de laquelle la productivité augmentera grâce à une meilleure collaboration entre les employés (et avec l'extérieur) et une plus grande participation.

4.3 Innovation permanente

Une autre grande promesse du SaaS est la mise à disposition beaucoup plus fréquente d'améliorations, de nouvelles fonctionnalités et de correction de bugs sans nécessiter de déploiement de nouvelles versions logiciels chez le client. Le rythme de mise à jour est en effet plus rapide avec les applications SaaS : plusieurs mises à jour par an contre 1 tous les 2 ans pour les applications sous licence hébergée (Boucher, 2009).

5 ENJEUX, FREINS ET LIMITES

Nous examinerons dans ce qui suit les freins à l'adoption du SaaS par les entreprises et les points sur lesquels elles devront porter leur vigilance quand elles auront recours au SaaS.

1. **Point de vue économique :** Si le modèle SaaS est économiquement intéressant lors de la mise en place de l'application, il peut s'avérer au final plus couteux. C'est ce que relève Gartner en précisant qu'à partir de la troisième année et en tenant compte des amortissements du matériel et des logiciels, une solution licence hébergée en interne peut s'avérer moins couteuse (Desisto, 2009) (Boucher, 2009).

2. **Point de vue fonctionnel : intégration et personnalisation :** deux problèmes sont fréquemment mis en avant :

   - L'intégration avec le système d'information de l'entreprise ou avec d'autres services SaaS (Boucher, 2009).
   - La personnalisation des applications SaaS aux besoins spécifiques de l'entreprise (Boucher, 2009) (Lu & Sun, 2009).

   Ces deux problèmes s'opposent à la promesse de simplicité et de rapidité de mise en œuvre du SaaS et sont de véritables obstacles aux gains de productivité que peuvent apporter les services SaaS.

3. **Point de vue de la sécurité** : Les applications SaaS cumulent les risques liés au service proprement dit, à l'externalisation et à la mutualisation.

   - Les risques liés au service sont ceux de tout service informatique comme la perte de données, la divulgation d'informations, la gestion des droits d'accès et la traçabilité (Boucher, 2009).
   - Un point de vigilance porte sur la gestion des accès (politique de mot de passe par exemple) qui est souvent héritée des applications grand public et ne satisfait pas aux exigences de la politique de sécurité en entreprise (Lu & Sun, 2009). Une bonne solution qui de plus améliore l'ergonomie en évitant la multiplication des mots de passe est d'utiliser le système d'authentification de l'entreprise pour l'accès au service de SaaS (principe de single sign on).
   - L'externalisation ajoute des risques autour de l'exposition aux attaques potentiellement plus importante, à la maîtrise des niveaux de service et de l'accessibilité du service.

4. **Confidentialité :** Au-delà de la sécurité des données et des accès, le problème du respect de la confidentialité des données de l'entreprise est très souvent avancée comme un frein à l'adoption de services SaaS (Boucher, 2009). C'est probablement l'aspect le plus délicat des résistances au SaaS dans le sens où il se réduit au final à la confiance que les décideurs de l'entreprise ont dans le prestataire.

6 CONCLUSION

L'étude comparative nous a permis de bien comprendre les différences entre les SaaS et les autres technologies et architectures afin de ne pas interchanger a tort ces différents concepts. Nous avons également présenté les avantages et inconvénients des Saas afin de saisir l'apport de cette nouvelle notion.

A travers quelques travaux de recherche, nous avons pu ressortir les problèmes actuels des SaaS tels que le problème de sélection et de la souscription. La prochaine section concernera cet aspect.

QUATREME CHAPITRE : DISCUSSION

1 INTRODUCTION

Comme nous l'avons présenté dans les chapitres précédents, le modèle SaaS est devenu très populaire ces derniers temps. Il est considéré comme un moyen pour baisser les coûts IT et accélérer les délais de livraison des produits. Cela est dû aux services « prêts-à-employer » (ready-to-use), l'infrastructure dynamiquement scalable et à la notion de payement en fonction d'utilisation (pay-per-use).

Bien que le SaaS soit un phénomène récent, une bonne quantité de recherche a été rapportée dans des différents secteurs tels que la configuration (Sun W. , Zhang, Jie, Sun, & Su, 2008), la sécurité (B.R. Kandukuri, 2009), l'intégration et la gestion de réseau (T. Miyamoto, 2009).

Nous nous intéressons plus particulièrement à la conception d'une couche intermédiaire qui gère la publication et t la sélection des produits SaaS ainsi que la souscription et les critères de choix des produits les plus appropriés à la demande des clients. Nous faisons donc une synthèse des travaux de recherche récents développés dans ce domaine.

Dans ce qui suit, nous présentons quelques travaux de recherche qui traitent principalement :

- Le problème de la nécessité d'un intergiciel pour les applications Saas,
- Le problème de la sélection adaptée des produits saas ainsi que les différent paramètres de choix, et
- Le problème de la souscription aux applications saas

2 LA NECESSITE D'UN SERVICE BROKER (SB)

C'est donc à cause de la nature autonome des fournisseurs d'applications SaaS, qu'il faut y avoir une sorte d'intergiciel « service broker » indépendant pour que toutes les applications puissent communiquer. Cette question de la nécessité d'un service broker a été traitée dans de nombreux articles récents (Turner, Budgen, & Brereton, 2003) (Bai, 2010) (Wittgreffe, 2010) (Stamford, 2009). Dans (Dhesiaseelan & Ragunathan, 2004), (Turner, et al., 2004), (Bennett & al., 2003) et (Howard & Kerschberg, 2004), les auteurs ont proposé des services broker pour résoudre des problèmes spécifiques. L'objectif du broker est initialement de prendre la grande partie de la surcharge de diverses tâches de communication pour créer un environnement de confiance :

- Dans (Tian M. , Gramm, Naumowicz, Ritter, & Schiller, 2003), les auteurs ont été préoccupés par l'intégration de la qualité de service (QoS) dans les services web pour une meilleure sélection fondée sur les exigences des clients. Ils ont introduit également une architecture Web Service Broker (WSB) qui contribuerait à l'accélération du processus de la recherche du service approprié. Cependant, le WSB avait plus de potentiel pour créer un environnement de confiance au lieu d'être juste en mesure d'accélérer la recherche du service web.

- Une autre approche intéressante a été décrite dans (Turner, et al., 2004) où les auteurs ont créé un service broker en utilisant des services web dans le cadre de la recherche des informations de gestion dans des environnements distribués. L'architecture proposée est composée de 3 services primaires : les services de sécurité (SS), les services d'intégration du broker (IBS) et un service d'accès aux données (DAS).

- B. Moore et Q. H. Mahmoud ((Moore & Qusay, 2009) ont présenté un service Broker pour les applications SaaS, qui aborde les préoccupations actuelles d'utilisation du Web en tant que plate-forme. Le service Broker proposé agit comme un conteneur pour la publication d'applications

SaaS hétérogènes des différents fournisseurs autonomes. B. Moore et Q. H. Mahmoud ont également démontré que le service Broker permet aux utilisateurs d'accéder aux applications SaaS et aux services web et à leurs données à partir d'un environnement de confiance ou de n'importe où dans le monde d'une façon transparente. Le service Broker proposé par (Moore & Qusay, 2009) s'occupe également des issues d'intégration et d'échange fiable de données provenant de différentes sources autonomes. (Moore & Qusay, 2009) ont essayé de résoudre principalement deux problèmes qui sont :

- Définir un environnement persistant qui permet aux applications SaaS d'être enregistrées, recherchées, souscrites et exécutées, et
- Construire un environnement persistant qui permet aux gens d'accéder à leurs ressources personnelles sur Internet.

Pour se faire, ils ont utilisé :

- La notion du WSDL étendu (Degwekar, Su, & Lam, 2004)pour la communication et le maping des données,
- Le service web comme technologie de support, et
- Les attributs d'interface tels que le coût, la disponibilité, les exigences de sécurité et la fiabilité pour mieux spécifier le produit SaaS et ainsi faciliter sa sélection et améliorer son utilisation.

Bien que, dans l'ensemble, ce travail explore toutes les questions relatives au médiateur, la recherche des applications qui répondent le mieux aux exigences des utilisateurs n'a pas été explicitement étudié.

3 LES PARAMETRES DE CHOIX D'UN PRODUIT SAAS

Quand plusieurs fournisseurs offrent des produits SaaS, le choix du produit devient une question principale. Cela nécessite une analyse des paramètres de choix et des offres des produits des différents fournisseurs. Quand des critères multiples sont impliqués dans la prise de décision, on parle d'un problème de la prise de décision multi-critères «Multi-Criteria Decision-Making» (MCDM) (Zeng, Benatallah, Lei, Mgu, Flaxer, & Chang, 2003).

Les problèmes de type multi-critères et multi-produits ne peuvent être résolus avec l'intuition ou par un simple jugement. Ces derniers peuvent fonctionner très bien, mais seulement quand le nombre des paramètres de choix est minimal. Durant le processus de sélection, les paramètres sont rangés habituellement en fonction de leur priorité. L'affectation des priorités aux produits implique de décider le poids de chaque paramètre. Tout en prenant en considération les poids des jugements, il est tout à fait probable que le jugement de l'utilisateur soit centré sur les paramètres principaux seulement. Ceci peut mener à une priorité inexacte et aux poids affectés aux paramètres incorrects. Pour prendre une décision, il est nécessaire d'avoir des valeurs quantifiables au lieu d'avoir des avis subjectifs.

Plusieurs travaux de recherches se sont focalisés sur la question :

- (Manish & Shrikant, 2009) ont proposé une approche pour aider les entreprises à choisir le produit SaaS le plus approprié à leurs besoins en se basant sur Analytic Hierarchy Process (AHP) développé par T. L. Saaty (Saaty & Vargas, 2001) (Saaty, 1990) pour l'analyse, l'extraction et la hiérarchisation des paramètres d'un logiciel.

Ils ont donc proposé différents facteurs de choix tels que la fonctionnalité, l'architecture, l'utilisabilité, la réputation de fournisseur et le coût. Le facteur de fonctionnalité inclut des attributs typiquement considérés en tant que modules fonctionnels de SF «sales force».
Le diagramme ci-dessous résume tous les facteurs du choix d'un produit SaaS proposé par (Manish & Shrikant, 2009) ainsi que les différents attributs.

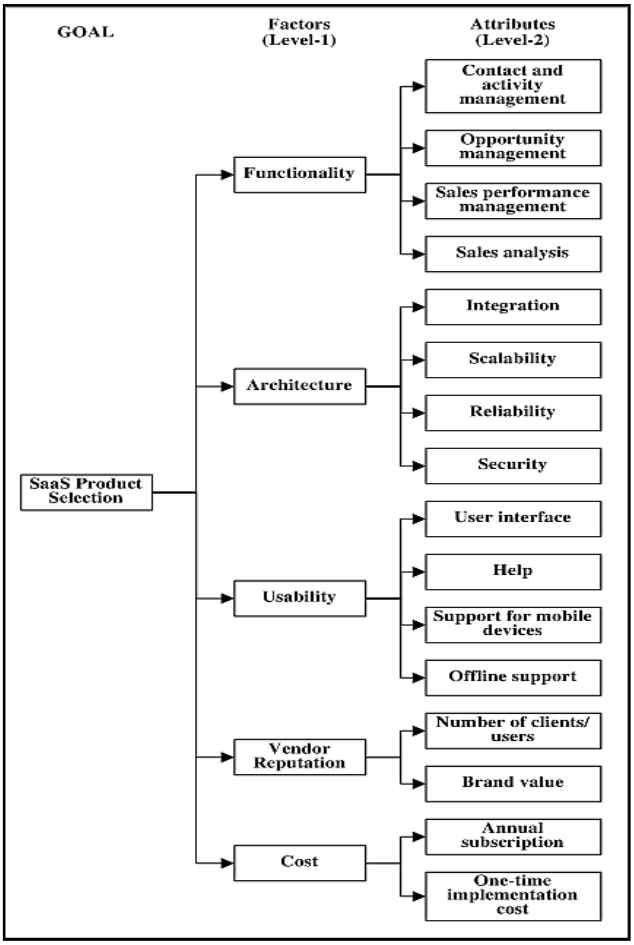


Figure 15. Caractéristiques et attributs de choix (Manish & Shrikant, 2009)

(Manish & Shrikant, 2009) ont commencé par la hiérarchisation des facteurs et des attributs du choix. L'avantage d'hiérarchiser est que le jugement peut être fait séparément pour chacune des multiples propriétés. Ceci est essentiel pour prendre une décision saine. Chaque élément est comparé par la suite à tous les autres éléments en lui attribuant un poids. Et de la même façon, tous les éléments sont comparés à chaque niveau de la hiérarchie.

A travers la lecture et l'analyse du travail de (Manish & Shrikant, 2009), nous avons trouvé que :

- Les paramètres de configurabilité et de personnalisation n'ont pas été pris en considération bien qu'ils soient des paramètres clés dans le processus du choix d'un logiciel SaaS.
- Leur approche ne peut être généralisable sur tous les types d'applications SaaS, et
- L'algorithme de calcul des poids, n'est pas explicitement mentionné dans l'article.

- Afin de servir efficacement un client spécifique, la plupart des logiciels doivent être plus ou moins adaptés. Les approches largement utilisées pour l'adaptation des logiciels sont : la configuration et la personnalisation. En fait, la configuration et de la personnalisation du logiciel ne sont pas de nouveaux issues. Il existe beaucoup de recherches universitaires et de pratiques

industrielles qui sont déjà faits dans cet axe. Parmi ces recherches nous trouvons : la théorie du management de la configuration de logiciel « Software Configuration Management (SCM) » qui a été développée par Roger Pressman dans le domaine de génie logiciel (Pressman, 1996-2001) (Sun, Zhang, Jie, Sun, & Su, 2008); les applications SAP ont aussi une grande capacité de personnalisation et de configuration en utilisant les GUI et les scripts basés sur les outils de programmation (ABAP) (Weiss, 2010).

Les fournisseurs des SaaS ont développé une profonde capacité de configuration ainsi de personnalisation. Par exemple, Salesforce.com fournit des Apex pour faciliter la configuration et la personnalisation de leurs larges applications sur le Web basées sur l'architecture multi-client (Salesforce.com). En outre, le modèle SaaS n'autorise pas le développement et la maintenance d'une version de code source indépendante pour chaque client. Alors, les fonctionnalités de personnalisation et de configuration jouent un rôle extrêmement important pour son succès.

En premier lieu, il faut savoir la différence entre la configuration et personnalisation, ainsi que les défis dans l'environnement multi-client de SaaS.

A. La configuration et la personnalisation dans un environnement multi-clients

La conception des SaaS est basée principalement sur les notions du service multi-client et unique-instance précédemment décrites. Cela veut dire que les fournisseurs de SaaS ne développent ni conservent différentes versions pour chaque client. Le cas idéal est de mettre à l'aise chaque client tout en utilisant une offre complètement standardisée. Cependant ce cas idéal habituellement ne se produit pas dans la zone des applications métiers. En général, plus la complexité fonctionnelle du logiciel augmente, plus les efforts potentiels d'adaptation doivent être mis en œuvre afin de pouvoir servir un client spécifique (Sun, Zhang, Jie, Sun, & Su, 2008).

Adapter un service SaaS peut exploiter deux approches principales : la configuration et la personnalisation. Dans ce qui suit, nous essayons de clarifier la différence entre ces deux concepts afin d'éviter toutes confusions. Comme illustré par la figure suivante, la configuration et la personnalisation peuvent prendre en charge l'adaptation à un certain niveau. Le cœur de la différence est la complexité.

- **La configuration** n'implique pas un changement de code source de l'application SaaS. Elle prend en charge généralement la variation en développant des paramètres prédéfinis, ou en tirant parti des outils conçus pour modifier les fonctions de l'application au sein d'une portée « scope » prédéfinie. Par exemple : ajouter des champs de données, changer les noms des champs, modifier les listes déroulantes, ajouter de boutons et modifier les règles métier,…etc. sont des fonctions prédéfinies. La configuration peut prendre en charge les exigences d'adaptation au sein d'une limite configurable prédéfinie.

- **La personnalisation** implique des modifications du code source de l'application SaaS pour créer une fonctionnalité qui est au-delà de la limite configurable.

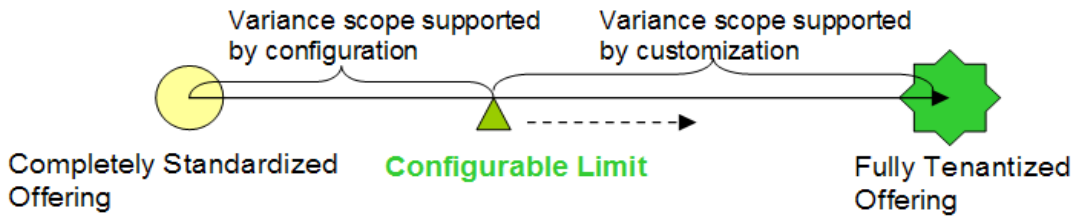


Figure 16. Configuration et personnalisation (Sun, Zhang, Jie, Sun, & Su, 2008)

La personnalisation est une approche beaucoup plus onéreuse par rapport à la configuration pour les fournisseurs de SaaS et les clients. Elle introduit des modifications du code source, ce qui produit des coûts importants tirés de nombreux issues tels que :

- Le besoin des personnes ayant un savoir et des compétences techniques plus élevées pour travailler sur la personnalisation ;
- L'allocation des ressources et de l'infrastructure pour la gestion et le stockage des différentes versions ;
- Un cycle de vie de développement, de débogage/test et de déploiement beaucoup plus long, etc. (Rohleder, Davis, & Günther, 2005).

La personnalisation devient beaucoup plus complexe dans le contexte des SaaS parce que les vendeurs doivent maintenir chaque morceau de code de personnalisation pour chaque locataire. La mise à niveau de l'application SaaS ne devrait pas conduire à perdre n'importe quel code de personnalisation. Les SaaS devraient donc éviter autant que possible la personnalisation en utilisant la configuration afin de répondre aux exigences d'adaptation. Pour cela, il faut également élargir la limite configurable autant que possible pour répondre aux exigences particulières du client.

B. Modèle de compétences

(Sun W. , Zhang, Jie, Sun, & Su, 2008) ont défini un modèle de compétences facilitant ainsi la définition et l'exécution d'une stratégie de personnalisation et de configuration des SaaS.

Level of Competency	Description	Approach	Variance Level Supported
Entry	Highly standardized offering without any configuration and customization support	Well design the functionalities as standardized offering to cover targeted customers	None
Aware	Relatively standardized offering with pre-defined variance points	Offer parameterized configuration	Low
Capable	Relatively standardized offering with user defined configuration	Offer self serve configuration tool to empower customers	Medium
Mature	Base offering with programable enviroment to enable user preferred customization	Offer scripting based programming for very flexible customization	High
World Class	Offer a platform supported by programming model and tools to enable extremely strong customization or even new application development	Offer well defined programming model and tools to enable extensive customization and new application development	Extremely High

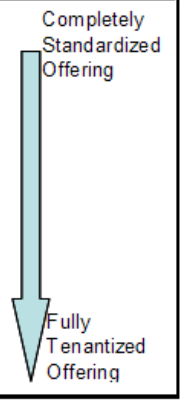


Tableau 5. Modèle de Compétence (Sun W. , Zhang, Jie, Sun, & Su, 2008)

- Comme c'est décrit dans le tableau ci-dessus, le modèle de compétences introduit par (Sun W. , Zhang, Jie, Sun, & Su, 2008) définit cinq niveaux de compétences qui sont : *Entry*, *Aware*, *Capable*, *Mature* et *World Class*.

- Différents niveaux de compétence peuvent permettre différents niveau de variation d'exigences.

- Les fournisseurs de SaaS peuvent avoir différentes stratégies en fonction des segments de la clientèle ciblée, la portée de la variation prise en charge, … etc

- Si la stratégie est bien définie, le service SaaS peut réussir sur le marché, même si ces compétences de configuration et de personnalisation restent uniquement au niveau «Entry » ou « Aware ».

La configuration et la personnalisation d'une application SaaS peuvent se produire dans de différentes perspectives. (Sun W. , Zhang, Jie, Sun, & Su, 2008) ont essayé d'analyser l'adaptation du point de vue exigences des clients. Les clients de SaaS peuvent avoir de potentielles exigences en matière de personnalisation et de configuration des différentes perspectives. Par exemple, chaque client peut exiger quelques changements tels que « J'ai besoin de plus de champs pour décrire mes documents d'entreprise» « Notre gestionnaire veut un nouveau rapport/dashboard afin d'analyser les données de ventes » ; « Notre organisation ne joue aucun rôle de gérant des achats » … etc.

(Sun, Zhang, Jie, Sun, & Su, 2008) ont introduit un modèle de compétences pour la configuration et la personnalisation et ainsi que les aspects qui sont habituellement exigés pour la configuration et la personnalisation. Ils ont présenté également une méthodologie pour guider les fournisseurs de SaaS à planifier et exécuter des stratégies de configuration et de personnalisation.

Cependant, une définition des différents aspects de la configuration et de la personnalisation ainsi que l'analyse détaillée n'ont pas été pris en considération dans le travaille de (Sun W. , Zhang, Jie, Sun, & Su, 2008).

4 LA SOUSCRIPTION A UN SAAS

Comme le SaaS adopte un modèle de tarification en fonction d'utilisation, le client doit être souscrit d'abord à un service afin de l'utiliser. Le fournisseur de services SaaS doit définir tous les services disponibles et comment s'abonner à chacun d'eux. De plus, le nombre de fournisseurs qui commencent à explorer le modèle SaaS est devenu de plus en plus important. Ces derniers introduisent différents types d'applications et des modèles métier (Ma & Seidmann, 2008).

- Dans l'industrie des télécommunications, les informations fournies à l'inscription sont utilisées comme une base de la gestion du cycle de vie d'un service. Par exemple, l'architecture de réseau des informations de télécommunications (Telecommunication Information Networking Architecture TINA[17]) spécifie un modèle d'information de gestion d'abonnement (d'inscription) pour gérer les relations et les interactions entre l'utilisateur du service, l'abonné et le détaillant[18]. Lee et al. fournissent un Framework CORBA basé sur trois niveaux pour la mise en œuvre d'un système de gestion des informations fournies par le service d'abonnement dans un environnement TINA (Lee, Mohapi, & Hanrahan, 2001). Concernant le système général de publication/abonnement dans l'industrie des télécommunications, la recherche et la pratique sont concentrées sur le processus de souscription au service et sur le modèle de données.

- La gestion d'abonnement pour les services Web englobe principalement l'émission des notifications d'événement et la gestion de la communication entre l'abonné et le Registre. Par exemple, « Universal Description Discovery and Integration (UDDI[19])» introduit un type d'entité appelé abonnement (subscription) dans son modèle d'information afin d'assurer le suivi des changements d'autres entités UDDI comme businessEntity, businessService, etc. L'UDDI définit une API dédiée à la gestion des 'interactions entre un abonné et le registre dans les modes synchrone et asynchrone. Dans (Liu, Huang, & Mei, 2006) et (Liu, Zhou, Huang, & Mei, 2007), Liu et al. introduisent une approche centrée consommateur pour la découverte de service d'une part

17 http://www.tinac.com/specifications/documents/comp.pdf
18 http://www.tinac.com/specifications/documents/overall.pdf
19 http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm

et l'abonnement au service le plus approprié d'autre part. Cette approche est basée sur le concept de communauté d'intérêt « Community-of-Interest (CoI) ». Cette communauté représente une collection de plusieurs services avec des fonctionnalités similaires et des qualités de service (QoS) différentes. Dans un modèle CoI, la communication entre un abonné et un service ne se fait pas directement, mais plutôt il interagit avec la CoI qui est responsable de la planification de l'abonnement dynamiquement avec les fournisseurs de services cibles afin d'atteindre le meilleur choix en fonction de plusieurs critères entre autre, la qualité de service.

- (Mietzner, Leymann, & Papazoglou, 2008), ont décrit un format de package pour les applications SaaS composites et configurables développées à base d'une architecture orientée service (SOA). Ils ont montré comment l'architecture de composants de service « service component architecture (SCA)» peut être étendu avec les descripteurs de variabilité et les modèles SaaS multi-clients en un package d'une part et déployer la propriété de multi-clients sur les composites configurables des applications SaaS d'autre part. Ils ont étudié la structure de service du point de vue configurabilité mais la souscription n'est pas explicitement décrite. La gestion d'abonnement pour les applications SaaS n'a donc pas été bien étudiée.

- (Jiang, Sun, Tang, Snowdon, & Zhang, 2009) ont introduit un modèle de souscription aux applications SaaS. Pour cela, ils ont commencé tout d'abord par la définition des entités et les relations entre elles. Une méthode a été donc développée pour faciliter à un fournisseur de SaaS, l'analyse de son application et le modèle métier ciblé afin d'aboutir à une bonne conception de la gestion de souscriptions. Cette méthode est basée sur deux types d'analyse :

  - **La structure de service** qui consiste à définir des différents éléments d'une offre de service ainsi que les relations entre eux. (Jiang, Sun, Tang, Snowdo, & Zhang, 2009) ont défini cinq patrons de structure de service qui sont : ***Single Service***, ***Independent Multiple Services***, ***Dependent Multiple Services***, ***Composed Service*** et ***Proxy Service.***

    Le processus de sélection d'un modèle de structure de service est illustré par la figure suivante comme un arbre de décision non exclusive.

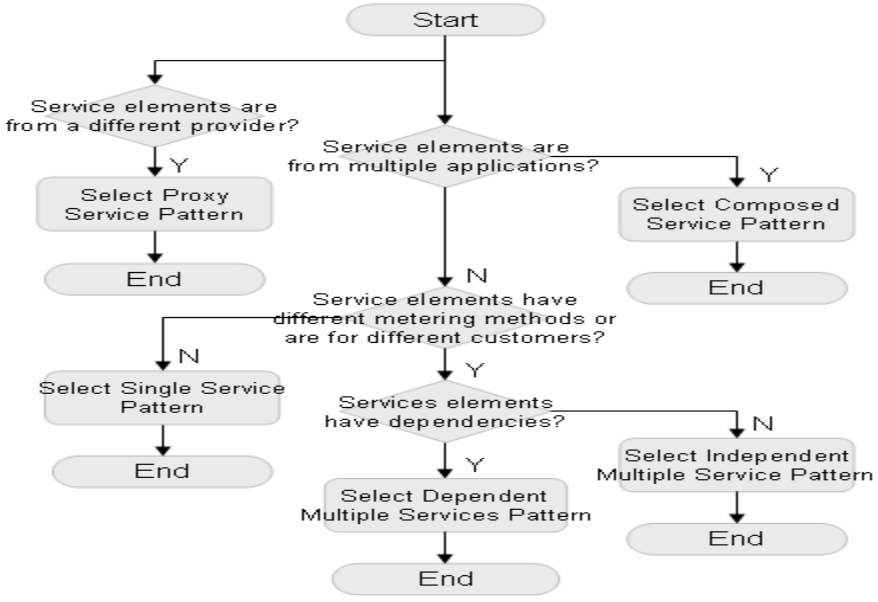


Figure 17. Arbre de décision des patrons de structure de service (Jiang, Sun, Tang, Snowdo, & Zhang, 2009)

Cet arbre peut être consulté plusieurs fois pour avoir une décision. Cependant, le problème qui se pose est que nous pouvons avoir plusieurs patrons pour la même offre de service. Par exemple, les deux patrons « Proxy » et « service composé » peuvent être sélectionnés en même temps.

- **Les interactions entre les applications métiers** : c'est l'analyse des interactions entre les applications métiers. (Jiang, Sun, Tang, Snowdo, & Zhang, 2009) ont défini quatre patrons d'interaction business qui sont : ***Self-service***, ***Hub-Spoke***, ***Distribution*** et ***Delegated.***

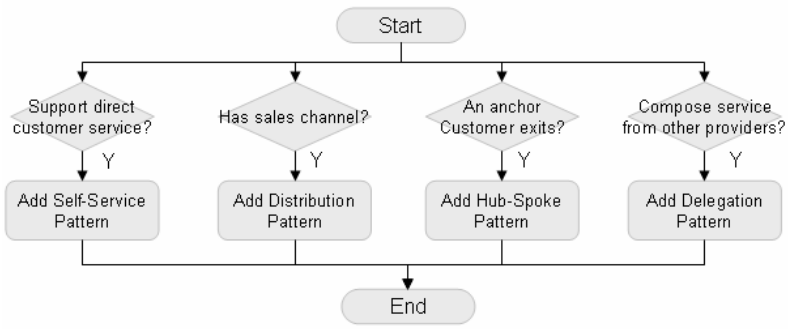


Figure 18. Arbre de décision des patrons d'interaction business (Jiang, Sun, Tang, Snowdo, & Zhang, 2009)

La sélection des patrons d'interaction business est illustrée par la figure suivante. Il s'agit d'un arbre de décision dégénéré ou tous les modèles peuvent être sélectionnés tant que les conditions proposées peuvent être satisfaites.

5 CONCLUSION

Cette recherche nous a permet de traiter quelques issues, quant à la nécessité d'avoir un service broker, la définition des paramètres permettant une sélection adaptée, et la définition d'un modèle de souscription aux applications SaaS.

L'exploration des travaux de chercheurs a démontré l'intérêt d'établir une couche intermédiaire pour les applications SaaS afin de :

- résoudre le problème d'intégration de ces derniers avec les autres plateformes,
- faciliter la communication et la coopération entre eux, et ;
- permettre leurs publications et leurs découvertes.

Et vu la prolifération des applications SaaS, il a été constaté que la recherche des applications qui répondent le mieux aux exigences des utilisateurs n'a pas été explicitement intégrée dans les médiateurs proposés.

Ainsi, une méthode de sélection adaptée d'un produit SaaS est basée sur la bonne définition des différents paramètres de choix. Nous avons trouvé que les paramètres de configurabilité et de personnalisation n'ont pas été étudiés bien qu'ils soient des paramètres clés dans le processus du choix d'un produit SaaS.

Vu que les SaaS sont consommés à travers le net, nous avons constaté aussi que la souscription à ces ses derniers est une étape suprême. C'est la raison pour la quelle des méthodes ont été développé afin d'aider les fournisseurs à mettre en œuvre leurs modèles de souscriptions.

Les prochaines chapitres seront consacrer a développer notre propre proposition, et qui consistera a :

- Proposer une méthode de sélection du Saas mieux adaptée aux exigences des clients. Pour cela, des définitions détaillées des différents paramètres du choix et l'algorithme sélection ont été établis.

- Proposer un modèle de souscription aux applications SaaS. Un modèle qui mise à l'amélioration de celui décrit par (Jiang, Sun, Tang, Snowdo, & Zhang, 2009). Nous avons simplifié ce composant pour avoir une décision plus précise du patron de souscription et nous avons proposé des expressions et un formalisme plus général, et

- Intégrer par la suite ces deux méthodes dans un médiateur pour les applications SaaS.

CINQUIEME CHAPITRE : APPROCHE PROPOSEE

1 INTRODUCTION

Tel que nous l'avons exploré dans le chapitre précédent, des recherches et des travaux ont traité quelques issus telles que la nécessité d'avoir un service broker, la définition des paramètres permettant une sélection adaptée, et la définition d'un modèle de souscription aux applications SaaS. Nous nous basons donc sur ces travaux pour faire et développer la conception d'un intergiciel (service broker) qui permettrais à la fois :

- Pour le fournisseur : la publication et la sélection d'un modèle de souscription relatives aux applications SaaS
- Pour le client : la recherche et la sélection du produit SaaS le mieux adapté.

Afin d'atteindre cet objectif et élargir les fonctionnalités d'un service broker :

- Nous proposons plusieurs facteurs pour le choix de SaaS tels que : la fonctionnalité, l'architecture, l'utilisabilité, la réputation du fournisseur, le coût et la configurabilité et la personnalisation.
- Nous esquissons un algorithme de sélection d'un SaaS
- Et nous proposons une méthode basée sur les patrons pour définir des modèles de souscriptions.

2 LA METHODE DE SELECTION D'UN SAAS LE PLUS PERTINENT

Lorsque les abonnés doivent prendre une décision « quel service choisir à partir d'une liste de résultats de recherche », il est bénéfique d'avoir le maximum d'informations possibles sur les services. Les registres UDDI actuels fournissent uniquement des informations générales sur les services tels que le fournisseur, l'emplacement et d'autres paramètres. L'UDDI ne contient pas toutes les informations pour aider l'abonné à distinguer un service d'un autre.

Dans le cadre du processus d'enregistrement, les fournisseurs peuvent faciliter leur enregistrement en utilisant « les Attributs d'Interface». C'est un formulaire web qui vise à aider ces derniers pour qu'ils puissent donner des attributs plus significatifs sur leur service web afin d'attirer de nouveaux abonnés en distinguant leurs services par rapport aux autres services du même type.

Ainsi, les application sont evaluées en se basant principalement sur ces attributs qui ne sont entre autre qu'un ensemble de modules fonctionnels et une liste des préférences. Cependant, les offres ne contiennent pas forcément tous les attributs de choix (définit dans la section suivante). De plus, les attributs n'ont pas nécessairement le même niveau de compétence. C'est la raison pour laquelle nous avons introduit un modèle de compétences afin d'évaluer les applications SaaS selon leurs capacités. Nous avons défini cinq niveaux de compétences qui sont : ***None, faible, moyen, fort et très fort.***

Ce modèle peut également être utilisé afin d'identifier les objectifs d'amélioration de l'application grâce à une analyse comparative avec le niveau de compétence des autres fournisseurs de service.

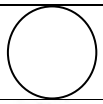
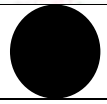
Le niveau	Description	Schéma	valeurs
None	L'offre X ne contient pas l'attribut		$V_{Ai/F}(X)=0$
faible	L'offre X contient l'attribut mais avec un niveau de compétence faible.		$V_{Ai/F}(X)=1$
Moyen	L'offre X contient l'attribut avec un niveau de compétence moyen.		$V_{Ai/F}(X)=2$
fort	L'offre X contient l'attribut avec un niveau de compétence fort.		$V_{Ai/F}(X)=3$
Très fort	L'offre X contient l'attribut avec un niveau de compétence très fort.		$V_{Ai/F}(X)=4$

Tableau 6. Modèle de compétence

Nous avons défini cinq valeurs différentes à savoir : 0, 1, 2, 3 et 4 pour chaque niveau de compétences. Ces valeurs vont être utilisées dans la section (2.3) pour le calcul des poids.

Dans ce qui suit, nous allons expliquer les paramètres et les attributs nécessaires pour une bonne sélection d'un produit SaaS ainsi qu'une méthode hiérarchique de calcul de priorité.

2.1 Le premier critère de choix : La fonctionnalité

Le facteur de fonctionnalité inclut des attributs typiquement considérés en tant que modules fonctionnels. Pour commencer, nous citons les deux catégories de logiciel d'application de SaaS les plus connus :

- La catégorie d'applications d'entreprise : c'est l'ensemble de logiciels qui répondent aux besoins des processus et de flux de données dans l'organisation, et souvent dans un environnement distribué. On trouve principalement, dans cette catégorie, les applications de : collaboration, gestion de contenu ou Content Management (CM), suite bureautique, CRM, Business Intelligence, ERP et la chaîne logistique étendue ou Supply Chain Management (Benlian, Hess, & Buxmann, 2009) (Weier & Smith, 2007) ; (Liao & Tao, 2008).
- La catégorie des logiciels **orientée-consommateurs :** offerts au grand public. (voir chapitre 2 section 2).

Notre travail se basera principalement sur la catégorie des applications d'entreprise et en utilisant le modèle de compétence, nous pouvons évaluer les différents facteurs de la fonctionnalité comme s'est montré dans le tableau 7 :

	None	Faible	Moyen	Fort	Très fort
Collaboration			2		
CM		1			
Suite bureautique					4
CRM					4
BI			2		
ERP		1			
SCM		1			

Tableau 7. Évaluation des différents facteurs de fonctionnalité

Dans ce qui suit, nous détaillons les modules fonctionnels de chaque classe d'applications métiers. Ces modules vont être pris en tant qu'attributs de choix.

2.1.1 La collaboration

Elle peut être divisée en trois fonctions (Lotus Development Corporation, 1995):

A. **Communication** : elle peut être considérée comme un échange d'informations non structurées. Par exemple, un appel téléphonique ou une discussion en message instantané (IM).
B. **Conférence :** se réfère à un travail interactif pour un objectif commun. Comme par exemple le Brainstorming [20]ou Remue-méninges.
C. **Coordination :** elle fait référence à un travail interdépendant complexe, fait dans le but de réaliser un objectif commun. Une bonne métaphore pour comprendre cela est de penser à une équipe de sports.

En utilisant le modèle de compétence précédemment décrit, nous pouvons évaluer les attributs du facteur de collaboration comme suit :

	None	Faible	Moyen	Fort	Très fort
Communication			2		
Conférence	0				
Coordination				3	

Tableau 8. évaluation des attributs de collaboration

Par exemple, on peut avoir une application X qui offre un module de communication qui a un niveau moyen et un autre de coordination qui a un niveau fort mais pas de conférence. Ainsi, on peut avoir $(3)^5$ applications de collaborations avec de différents niveaux de compétence.

2.1.2 La gestion de contenu (Content Management : CM)

Elle vise à gérer l'ensemble des contenus d'une organisation. Il s'agit de prendre en compte sous forme électronique les informations qui ne sont pas structurées, comme les documents électroniques, par opposition à celles déjà structurées dans les bases de données. Elle consiste souvent à réaliser les fonctions suivantes :

A. **La création** : c'est la fonction qui permet la création et l'édition de contenu.
B. **L'édition** : consiste à la mise au point du contenu du message et le style de la livraison, y compris la traduction et localisation.
C. **La publication :** c'est la livraison du contenu pour l'utilisation.
D. **L'administration :** chargée de gérer les autorisations d'accès aux dossiers et fichiers. Généralement accomplis en attribuant des droits d'accès aux groupes d'utilisateurs ou de rôles.

Nous évaluons les attributs du facteur CM en utilisant le même modèle de compétence

20 Le Brainstorming est une technique de résolution créative de problème sous la direction d'un animateur. http://fr.wikipedia.org/wiki/Brainstorming#cite_note-0

	None	Faible	Moyen	Fort	Très fort
La création			2		
L'édition	0				
La publication		1			
L'administration					4

Tableau 9. Évaluation des attributs du CM

De la même façon, on peut avoir 4^5 applications de gestion de contenu différentes.

2.1.3 Office software ou la suite bureautique

C'est un ensemble de logiciels qui informatisent les travaux courants dans un bureau et qui comporte typiquement :

A. **Le traitement de texte** : comme le Microsoft office Word
B. **Les Tableurs grapheurs** : comme le Microsoft office Excel
C. **La présentation** : comme le Microsoft office PowerPoint

Le tableau suivant présente l'évaluation des attributs de la suite bureautique basé sur le modèle de compétences.

	None	Faible	Moyen	Fort	Très fort
Traitement de texte			2		
Tableurs grapheurs	0				
Présentation					4

Tableau 10. évaluation des attributs de la suite bureautique.

On peut avoir 3^5 applications bureautiques différentes.

2.1.4 CRM (Customer Relationship Management)

Elle prend en charge un large éventail d'activités pour acquérir, renforcer ou conserver des clients. La fonctionnalité d'un logiciel CRM varie d'un vendeur à un autre. Quelques modules sont communs à la plupart des logiciels CRM qui sont :

A. **Module de Marketing direct** : permet aux entreprises d'identifier les clients cibles, générer des mails en direct et analyser la réponse de la clientèle cible.
B. **Module de ventes CRM** : implémente les fonctions de support avant-vente, la prise et l'ordonnancement des commandes, l'expédition et la facturation.
C. **Module de centre d'appels** : Un centre d'appels se compose d'une infrastructure de télécommunications complexes, d'un système informatique sophistiqué et des représentants des services spécialisés organisés pour gérer efficacement les appels téléphoniques entrants et sortants. Un module de centre d'appels CRM capte la grande quantité de données provenant des opérations du centre d'appel et les appels directement entrants aux représentants des services appropriés.
D. **Module d'aide de dépannage** : peut améliorer la satisfaction de la clientèle et la productivité grâce à l'automatisation des processus de support client.

Le tableau 11 illustre un exemple d'évaluation d'une application CRM qui a un niveau moyen du module de marketing direct, un centre d'appels avec un niveau faible et un aide de dépannage avec un niveau très fort.

	None	Faible	Moyen	Fort	Très fort
Marketing direct			2		
Ventes CRM	0				
Centre d'appels		1			
Aide de dépannage					4

Tableau 11. évaluation des attributs du CRM

On peut avoir $(4)^5$ applications CRM différentes.

2.1.5 L'informatique décisionnelle (Business Intelligence BI)

Désigne les moyens, les outils et les méthodes qui permettent de collecter, consolider, modéliser et restituer les données, matérielles ou immatérielles, d'une entreprise en vue d'offrir une aide à la décision et de permettre aux responsables de la stratégie d'entreprise d'avoir une vue d'ensemble de l'activité traitée. Un système d'information décisionnel (SID) assure quatre fonctions fondamentales, à savoir la collecte, l'intégration, la diffusion et la présentation des données. À ces quatre fonctions, s'ajoute une fonction de contrôle du SID lui-même, l'administration.

A. **La collecte** (parfois appelée *data pumping*) est l'ensemble des tâches consistant à détecter, à sélectionner, à extraire et à filtrer les données brutes issues des environnements pertinents compte tenu du périmètre du SID.
B. **L'intégration** consiste à concentrer les données collectées dans un espace unifié, dont le socle informatique essentiel est l'entrepôt de données. Élément central du dispositif, il permet aux applications décisionnelles de bénéficier d'une source d'information commune, homogène, normalisée et fiable, susceptible de masquer la diversité de l'origine des données.
C. **La diffusion (ou distribution)** met les données à la disposition des utilisateurs, selon des schémas correspondant au profil ou au métier de chacun,
D. **Présentation** Cette quatrième fonction, la plus visible pour l'utilisateur, régit les conditions d'accès de l'utilisateur aux informations. Elle assure le fonctionnement du poste de travail, le contrôle d'accès, la prise en charge des requêtes, la visualisation des résultats sous une forme ou une autre.
E. **Administration** C'est la fonction transversale qui supervise la bonne exécution de toutes les autres. Elle pilote le processus de mise à jour des données, la documentation sur les données (les métadonnées), la sécurité, les sauvegardes, la gestion des incidents.

L'évaluation des attributs d'une offre BI est basée sur le modèle de compétence comme le montre le tableau suivant :

	None	Faible	Moyen	Fort	Très fort
Collecte			2		
L'intégration		1			
diffusion				3	
Présentation			2		
administration					4

Tableau 12. évaluation des attributs du BI

On peut avoir $(5)^5$ applications BI différentes.

2.1.6 ERP (Enterprise Resource Planning)

Elle se compose de plusieurs modules fonctionnels. Ces modules fonctionnels imitent respectivement le domaine fonctionnel de l'organisation. Certains des modules fonctionnels du logiciel ERP sont comme suit :

A. **Finances** : C'est un module important puisque les autres modules sont intégrés dans les États financiers. Les bilans, comptes à payer, les débiteurs, budget, gestion de la trésorerie, comptabilité, etc. sont inclus dans ce module. C'est un module de base pour toute organisation.
B. **Planification de la Production :** le module de planification de la production permet au gestionnaire de planifier le travail de façon idéale. il gère l'utilisation optimale des ressources humaines et non humaines et l'ordonnancement de la production.
C. **Contrôle des stocks** : Il permet de maintenir les niveaux requis de stocks dans l'entrepôt. Il aide à identifier les exigences de l'inventaire, fournit des options de réapprovisionnement, rapporte l'état de l'inventaire, surveille l'utilisation d'éléments, définit les objectifs, etc.
D. **Ressource humaine** : Ce module est également largement implémenté. Il simplifie la gestion des ressources humaines. Il maintient une base de données d'employé ; coordonnées, fréquentation, salaire, détails de promotion, évaluation du rendement, etc.. Cette information peut être utilisée pour optimiser l'expertise des employés.
E. **Achat** : Le module d'achat permet de simplifier l'acquisition des matières premières nécessaires. Les divers processus impliqués comme prix de négociation, identifier les fournisseurs potentiels, attribution de bon de commande et les processus de facturation sont automatisées. Le module de contrôle de l'inventaire et le module de planification de la production est intégré avec ce module.
F. **Les ventes** : le module des ventes met en œuvre les fonctions de l'ordre de planification, de placement, d'expédition et de facturation.

Le tableau suivant représente une évaluation basée sur le modèle de compétence d'une application ERP.

	None	Faible	Moyen	Fort	Très fort
Finances		1			
Planification de la Production	0				
Contrôle des stocks				3	
Ressource humaine					4
Achat		1			
ventes			2		

Tableau 13. Évaluation des attributs du ERP

On peut avoir 6^5 applications ERP différentes.

2.1.7 La chaîne logistique étendue (Supply Chain Management : SCM)

Elle désigne les outils et méthodes visant à améliorer et automatiser l'approvisionnement en réduisant les stocks et les délais de livraison. On parle ainsi de travail en "flux tendu" dont le but est d'accélérer les flux de matières dans l'entreprise. Une caractéristique majeure est donc l'ajustement des stocks sur toutes les chaînes logistique et de production afin de limiter le temps d'écoulement entre le fournisseur et le client final. Il s'agit de définir de manière optimale les quantités à acheter auprès des différents fournisseurs, produire dans les usines, stocker dans les entrepôts puis distribuer vers les clients.

Le dictionnaire APICS[21] définit la SCM comme suit : "design, planning, execution, control, and monitoring of supply chain activities with the objective of creating net value, building a competitive infrastructure, leveraging worldwide logistics, synchronizing supply with demand and measuring performance globally."

Les activités opérationnelles les plus importantes d'une SCM sont :

A. La production quotidienne et la planification de la distribution, y compris tous les nœuds dans la chaîne d'approvisionnement.
B. L'ordonnancement de la production pour chaque unité de production dans la SCM
C. La demande de planification et de prévision : coordonner les prévisions de la demande de tous les clients et de partage les prévisions avec tous les fournisseurs.
D. Les opérations entrantes, y compris la réception de l'inventaire.
E. Les opérations sortantes, y compris toutes les activités d'achèvement, l'entreposage et le transport aux clients.

L'évaluation des attributs d'une offre SCM basé sur le modèle de compétence se fait de la façon suivante :

	None	Faible	Moyen	Fort	Très fort
Production quotidienne			2		
Ordonnancement	0				
Planification et prévision				3	
Opérations entrantes				3	
Opérations sortante					4

Tableau 14. Évaluation des attributs du SCM

On peut avoir 5^5 applications SCM différentes.

2.2 Deuxième critère de choix : la liste des préférences

Beaucoup de facteurs sont impliqués dans le choix d'un produit de logiciel. En nous basant sur l'expérience et les avis des experts, nous proposons plusieurs facteurs pour le choix de SaaS tels que : l'architecture, l'utilisabilité, la réputation de fournisseur, le coût, la configurabilité et la personnalisation.

	None	Faible	Moyen	Fort	Très fort
Réputation			2		
Coût		1			
Utilisabilité				3	
Architecture			2		
Configurabilité et personnalisation					4

Tableau 15. Evaluation des différents facteurs de la liste des préférences

[21] APICS The Association for Operations Management is the global leader and premier source of the body of knowledge in supply chain and operations management, including production, inventory, materials management, purchasing, and logistics[http://www.apics.org/About/]

2.2.1 La réputation du fournisseur

Le facteur de réputation de fournisseur a été étudié dans quelques articles récents (Limam & Boutaba, 2010) (Yu, Liu, Hu, & Lin, 2009) (Eisentraut, Koch, & Mösle, 2001). Il inclut deux attributs :

- **Le nombre de clients/users** exprime le niveau de l'utilisation, qui indique si le produit est assez nouveau par exemple ou encore s'il est bien établi.
- **La valeur de marque du fournisseur** est également importante, car parfois un produit de fournisseur bien connu peut être préféré à un produit dont le fournisseur est non connu.

L'évaluer des offres par rapport à ce facteur est donnée par le tableau suivant :

	None	Faible	Moyen	Fort	Très fort
Le nombre de clients				3	
La valeur de la marque			2		

Tableau 16. Evaluation des attributs de la réputation du fournisseur

Une application « X » peut avoir un nombre élevé d'utilisateurs et une moyenne valeur de marque.

2.2.2 Le coût

Le facteur du coût inclut deux attributs : **abonnement annuel** et **coût d'implémentation**. Généralement, le coût du matériel et le support personnel sont couverts sous l'abonnement annuel, alors que le coût de consultation, d'efforts de configuration, etc. est couvert sous le coût d'implémentation.

De la même façon précédente, nous avons utilisé le modèle de compétence pour évaluer les attributs du facteur coût mais on a changé l'ordre les valeurs pour qu'il soit décroissant. Ainsi, le niveau None a comme valeur 4, le niveau faible 3 et ainsi de suite jusqu'au niveau très fort qui reçoit la valeur 0.

	None	Faible	Moyen	Fort	Très fort
Abonnement annuel		1			
Coût d'implémentation				3	

Tableau 17. Evaluation des attributs de coût

Une application « Y » peut avoir un coût d'a**bonnement annuel** faible et un coût d'implémentation fort.

2.2.3 L'utilisabilité

Les attributs reliés à l'**utilisabilité** sont comme suit :

- **La fiabilité** se rapporte à la capacité du produit SaaS de rester disponible pour ses utilisateurs pour des slots de temps défini. Il est nécessaire alors de déployer la surveillance et les outils diagnostiques ;
- **Interface utilisateur ergonomique** inclut des facettes qui assurent différents facteurs tels que l'intuitivités, la facilité d'utilisation pour les tâches fréquentes et nature esthétique des éléments graphiques.
- **L'attribut d'aide** se rapporte à la disponibilité des manuels faciles à utiliser, des modules e-learning, et de l'aide sensible au contexte.
- **Le support pour dispositif mobile** est devenu important pendant que la main d'œuvre moderne de vente dépend intensivement des dispositifs mobiles tels que PDA etc.
- **Le support « off line »** est devenu important aussi. Il signifie que les produits SaaS soutiennent un mécanisme laissent des utilisateurs travailler sur le système en mode déconnecté et les synchroniser une fois relié à l'Internet.

- **L'attribut d'intégration :** c'est la capacité d'une application à s'intégrer avec d'autres. L'attribut d'intégration devient tout à fait approprié pour les produits SaaS car ces derniers sont accueillis off-premise. Par conséquent, il paraît difficile de les intégrer avec les systèmes on-premise.

Le tableau 18 illustre la façon d'évaluer les attributs de l'**utilisabilité** à l'aide du modèle de compétence définit.

	None	Faible	Moyen	Fort	Très fort
Fiabilité			2		
Interface ergonomique		1			
L'attribut d'aide				3	
Support pour dispositif mobile	0				
Le support « off line »	0				
Intégration	0				

Tableau 18. Évaluation des attributs d'utilisabilité

Une application X peut être moyennement fiable, possédant une faible interface ergonomique et un fort support « off line ».

2.2.4 L'architecture

Les attributs d'une architecture SaaS sont détaillés dans le chapitre II, et qui sont :

- Le paradigme **Multi-client**
- La **scalabilité** ou la monté à l'échelle : se rapporte à la capacité du produit SaaS de maintenir un temps de réponse raisonnable pour ces utilisateurs même pendant la charge maximale. On peut définir deux sous-attributs : **la monté à l'échelle des données** et **la monté à l'échelle de l'application.**
- **La sécurité** est considérée comme le souci principal pour des produits SaaS. Les fournisseurs ayant des certifications tels que d'ISO 27000 assurent la sécurité de la manipulation des données des clients. Nous pouvons définir deux sous-attributs qui sont **l'authentification** et **l'autorisation.**

Le tableau suivant présente l'évaluation des attributs de la suite bureautique basé sur le modèle de compétences.

	None	Faible	Moyen	Fort	Très fort
Multi-client			2		
Scalabilité	0				
Sécurité					4

Tableau 19. Evaluation des attributs de l'architecture

2.2.5 Configurabilité et personnalisation

Il peut y avoir différents aspects dans lesquels un logiciel SaaS peut être configuré. Les sous-sections suivantes explorent la configurabilité du logiciel SaaS dans différents aspects. (Sun, Zhang, Jie, Sun, & Su, 2008) (Nitu, 2009). Nous allons utiliser ses aspects en tant qu'attributs du facteur «configuration et personalisation» .

- **Structure de données** : Il est possible de fournir un modèle pour le stockage des données qui répond aux exigences les plus communes des client avec une option d'ajouter des exigences de données spécifiques pour chaque client telle que : ajouter des champs à une table, ajouter une table supplémentaire ou des contraintes. Ainsi, la configurabilité des données est nécessaire

pour répondre à l'extensibilité dans le modèle de données tel qu'il est requis par les clients. (Sun, Zhang, Jie, Sun, & Su, 2008) (Nitu, 2009).

- **Interface utilisateur IHM** : La configurabilité d'interface utilisateur désigne la capacité de modifier l'aspect et la convivialité de cette dernière. Un client peut configurer l'interface en modifiant son apparence afin de refléter le produit corporatif(Chong & Carraro, 2006) (Sun, Zhang, Jie, Sun, & Su, 2008) (Nitu, 2009).

- **Workflow ou business process**, de plus que la configuration de l'interface utilisateur, qui donne une apparence unique pour les applications SaaS, il faut que le comportement de l'application puisse être configuré. Ceci est important parce que le même type de workflow peut avoir des comportements différents dans des organisations différentes. Le flux de travail se compose de l'ensemble d'activités, des rôles et de règles métier (Chong & Carraro, 2006) (Sun, Zhang, Jie, Sun, & Su, 2008) (Nitu, 2009).

- **Structure organisationnelle**, c'est l'ensemble des rôles et des entités de l'organisation (Sun, Zhang, Jie, Sun, & Su, 2008).

- **Contrôle d'accès** : Les clients des applications SaaS peuvent avoir plusieurs utilisateurs. Donc, ils sont responsables de la création des comptes individuels pour les utilisateurs finaux et la détermination les ressources et les fonctionnalités auxquelles chaque utilisateur devrait être autorisé à y accéder. Pour une organisation, puisqu'il y a toujours des données spécifiques de contrôle d'accès, il est nécessaire que le concepteur soit capable de créer, modifier ou supprimer des rôles ou des utilisateurs de son organisation (Chong & Carraro, 2006) (Sun, Zhang, Jie, Sun, & Su, 2008) (Nitu, 2009).

Comme nous l'avons indiqué dans le tableau 6, il existe de différentes approches permettant de différents niveaux de compétence. Nous avons présenté l'évaluation des applications par rapport à la configurabilité dans le tableau suivant :

	None	*Faible*	*Moyen*	*Fort*	*Très fort*
Structure de données		1			
IHM			2		
Workflow				3	
Structure organisationnelle	0				
Contrôle d'accès					4

Tableau 20. Évaluation des attributs de collaboration

Une application X peut avoir un niveau de configuration de structure de données faible, un IHM moyen, un flux de travail fort et un contrôle d'accès très fort.

Pour récapituler, nous résumons tous les facteurs et attributs définis dans cette section à travers un diagramme (voir figure 19).

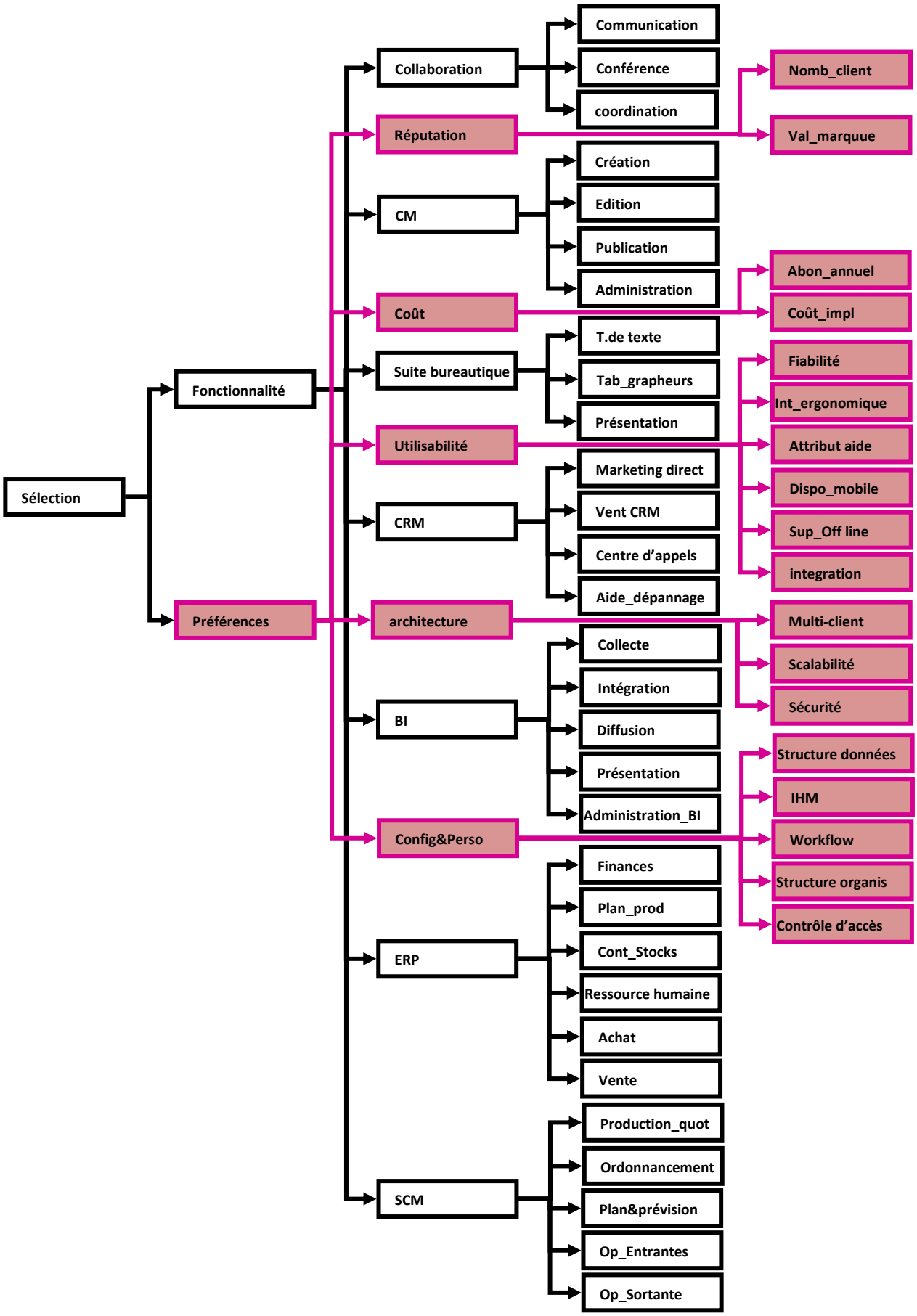


Figure 19. Diagramme récapitulatif des différents facteurs et attributs

2.3 L'algorithme de sélection d'un produit SaaS

La méthodologie adoptée débute par la compréhension des paramètres satisfaisants les exigences de l'application. Deux types d'outils d'AHP sont développés à partir de cette hiérarchie pour la comparaison par paires. L'un est conçu pour la comparaison des paramètres et l'autre pour la comparaison des produits.

La méthodologie que nous adoptons pour faire le choix d'un produit SaaS est divisée en trois parties :

- La première partie couvre le calcul des poids des attributs en fonction des priorités vis-à-vis les utilisateurs.
- La deuxième partie est faite pour calculer les paramètres globaux des attributs des produits.
- La troisième partie combine les résultats obtenus à partir des deux premières pièces pour ranger les produits.

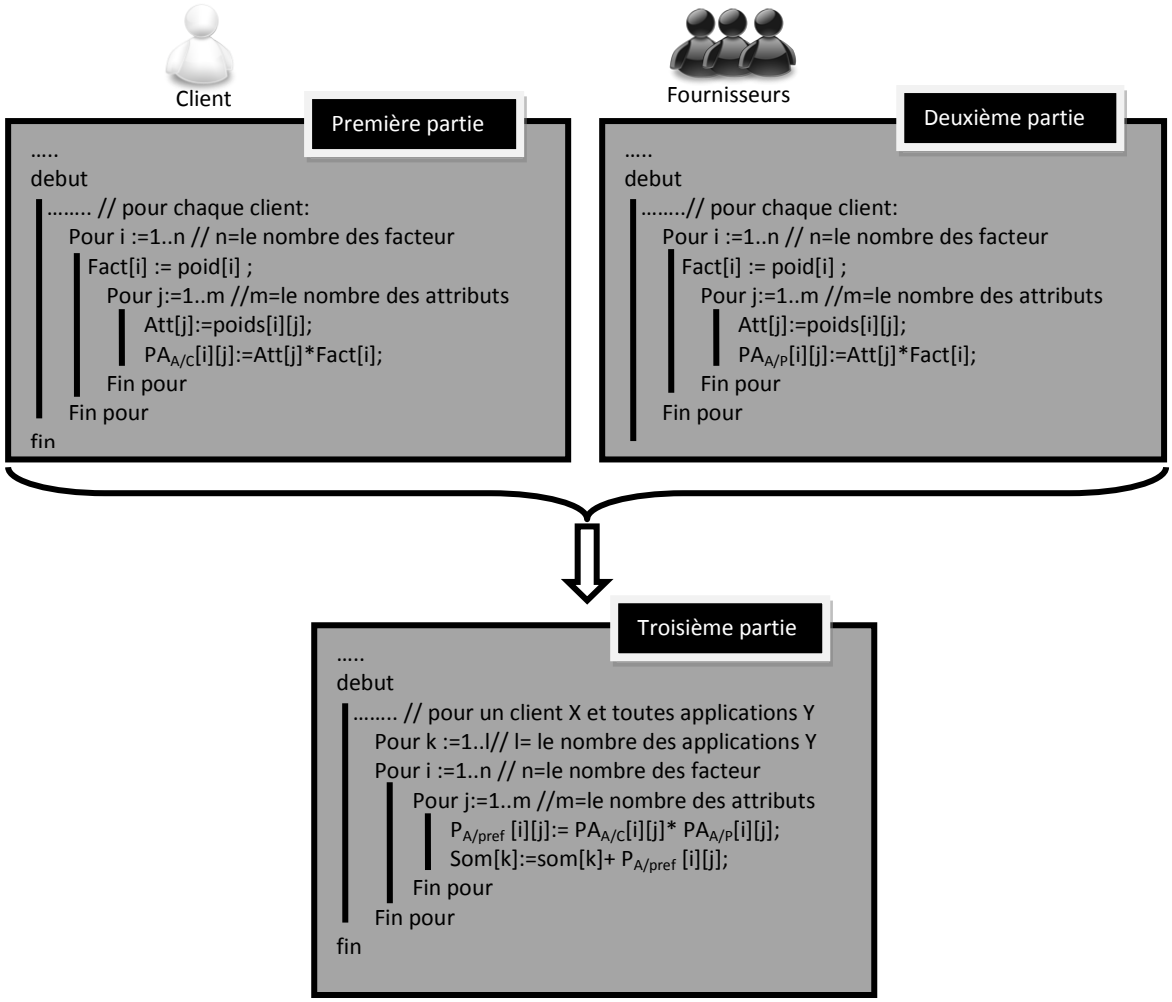


Figure 20. Schéma global de l'algorithme de sélection

Nous présentons en détail les trois étapes de notre algorithme dans les sections suivantes.

2.3.1 La première partie

Pour calculer le poids de chaque attribut selon les préférences d'un utilisateur, il faut :

1. Affecter des poids aux différents facteurs situé ci-dessus en fonction des préférences tel que : $\sum P_F = 1$
2. Affecter des poids selon les préférences aux différents attributs de chaque facteur $P_{A/F}$ tel que : $\sum P_{Ai/F} = 1$ / i= i…n avec n : le nombre des attributs/facteurs.
3. Calculer le poids effectif de chaque attribut selon les préférences du client de la façon suivante : $PA_{A/C} = P_{A/F} * P_F$

2.3.2 La deuxième partie

Pour calculer le poids de chaque attribut d'un produit SaaS donné, il faut :

1. Affecter des poids aux différents facteurs situait ci-dessus pour chaque produit SaaS tel que : $\sum P_F = 1$
2. Affecter des poids aux différents attributs de chaque facteur d'un produit SaaS $P_{A/F}$ tel que : $\sum P_{Ai/F} = 1$ / i= i..n avec n : le nombre des attributs/facteurs.
3. Calculer le poids effectif de chaque attribut d'un produit SaaS de la façon suivante : $PA_{A/P} = P_{A/F} * P_F$

2.3.3 La troisième partie

Pour savoir quel est le produit le plus approprié aux exigences d'un client X, il faut calculer la somme des attributs de chaque produit en fonction des préférences du client de la façon suivante :

1. Pour chaque produit SaaS, on calcule le poids $P_{A/pref}$ de chaque attribut selon les préférences d'un client X de la façon suivante : $P_{A/pref} = PA_{A/C} * PA_{A/P}$
2. On calcule la somme des $P_{A/pref}$ pour chaque produit, le résultat sera le poids général du produit SaaS P_P : $P_P = \sum P_{Ai/pref}$ tel que i=1…n et n est le nombre de tous les attributs.
3. Le produit SaaS le plus approprié aux exigences du client sera celui qui a le P_P le plus grand.

2.3.4 Le calcul des poids des différents attributs

Pour une application « X », le calcul des poids des attributs est donné par la formule suivante :

$P_{Ai/F}(X) = V_{Ai/F}(X) / \sum V_{Aj/F}(X)$;

Pour que ça soit plus clair, on va prendre l'exemple de l'application X par rapport au premier facteur qui est « Collaboration ». Pour cela, on doit calculer la somme des valeurs des attributs par rapport à ce facteur, donc :

$\sum V_{Aj/F}(X) = 2+0+3 = 5$

Donc :

	Poids
Communication	$P_{A1/F}(X) = \mathbf{2/5}$
Conférence	$P_{A2/F}(X) = \mathbf{0/5}$
Coordination	$P_{A3/F}(X) = \mathbf{3/5}$

2.3.5 Le calcul des poids des différents facteurs

De la même façon, le calcul des poids des facteurs est donné par la formule suivante :

$P_{Fi}(X) = V_{Fi}(X) / \sum V_{Fi}(X)$;

Pour cela, nous avons calculé les poids des facteurs de l'application X illustrée par le tableau 15, donc :

$\sum V_{Fi}(X)=2+1+3+2+4=10$

	Poids
Réputation	$P_{F1}(X)=2/12$
Coût	$P_{F2}(X)=1/12$
Utilisabilité	$P_{F3}(X)=3/12$
Architecture	$P_{F4}(X)=2/12$
Configurabilité et personnalisation	$P_{F5}(X)=4/12$

2.4 Exemple récapitulatif

Dans cette section, nous présentons un exemple illustratif de l'emploie de notre algorithme de sélection pour le choix d'une application CRM avec un niveau de compétences très fort. L'offre recherchée doit avoir de plus des modules de collaboration avec un niveau de compétences moyen. Il doit avoir aussi la liste des préférences suivante : un niveau de réputation moyen, coût faible, une forte utilisabilité, une architecture moyenne et un très fort niveau de configuration et de personnalisation.

	Facteurs	$P_{Fi}(X)$	attribut	$P_{Ai/F}(X)$	$PA_{A/C}$
Fonctionnalité	CRM	4/6	Marketing direct	2/7	8/42
			Ventes CRM	0/7	0/42
			Centre d'appels	1/7	4/42
			Aide de dépannage	4/7	16/42
	Collaboration	2/6	Communication	2/5	4/30
			Conférence	0/5	0/30
			Coordination	3/5	6/30
Préférences	Réputation	2/12	Le nombre de clients	3/5	6/60
			Valeur de la marque	2/5	4/60
	Coût	1/12	Abonnement annuel	1/4	1/48
			Coût_implémentation	3/4	3/48
	Utilisabilité	3/12	Fiabilité	2/6	6/72
			Int_ergonomique	1/6	3/72
			L'attribut d'aide	3/6	9/72
			Sup_dispositif mobile	0/6	0
			Le support « off line »	0/6	0
			Intégration	0/6	0
	Architecture	2/12	Multi-client	2/6	4/72
			Scalabilité	0/6	0
			Sécurité	4/6	8/72
	Configurabilité et personnalisation	4/12	Structure de données	1/10	4/120
			IHM	2/10	8/120
			Workflow	3/10	12/120
			Struct_organisationnelle	0/10	0
			Contrôle d'accès	4/10	16/120

Tableau 21. Les poids de chaque attribut selon les préférences d'un utilisateur

Le tableau 21 montre les différent facteurs et attributs de l'offre recherchée ainsi que le poids effectif de chaque attribut.

	Facteurs	attribut	Offre A ($PA_{A/P}$)	Offre B ($PA_{A/P}$)	Offre C ($PA_{A/P}$)
Fonctionnalité	CRM	Marketing direct	6/30	9/36	16/84
		Ventes CRM	4/30	3/36	12/84
		Centre d'appels	2/30	3/36	8/84
		Aide de dépannage	0	12/36	12/84
	Collaboration	Communication	3/15	3/24	12/70
		Conférence	3/15	0	6/70
		Coordination	3/15	3/24	12/70
Préférences	Réputation	Le nombre de clients	0	4/33	9/105
		Valeur de la marque	0	2/33	12/105
	Coût	Abonnement annuel	2/18	1/33	6/90
		Coût_implémentation	1/18	2/33	12/90
	Utilisabilité	Fiabilité	0	6/88	6/255
		Interface ergonomique	6/24	2/88	4/255
		L'attribut d'aide	3/24	6/88	6/255
		Sup_dispositif mobile	3/24	0	6/255
		Le support « off line »	0	2/88	4/255
		Intégration	0	0	8/255
	Architecture	Multi-client	1/24	4/66	9/135
		Scalabilité	2/24	0	6/135
		Sécurité	1/24	8/66	12/135
	Configurabilité et personnalisation	Structure de données	2/24	4/110	8/240
		IHM	1/24	4/110	12/240
		Workflow	0	12/110	16/240
		Struct_organisationnelle	0	4/110	12/240
		Contrôle d'accès	1/24	16/110	16/240

Tableau 22. Les poids de chaque attribut des offres SaaS

Le tableau 22 illustre l'exemple de trois offres existants : A, B et C. nous allons appliquer notre algorithme pour savoir, parmi ces trois produit, le quel qui répond le mieux aux exigences du client.

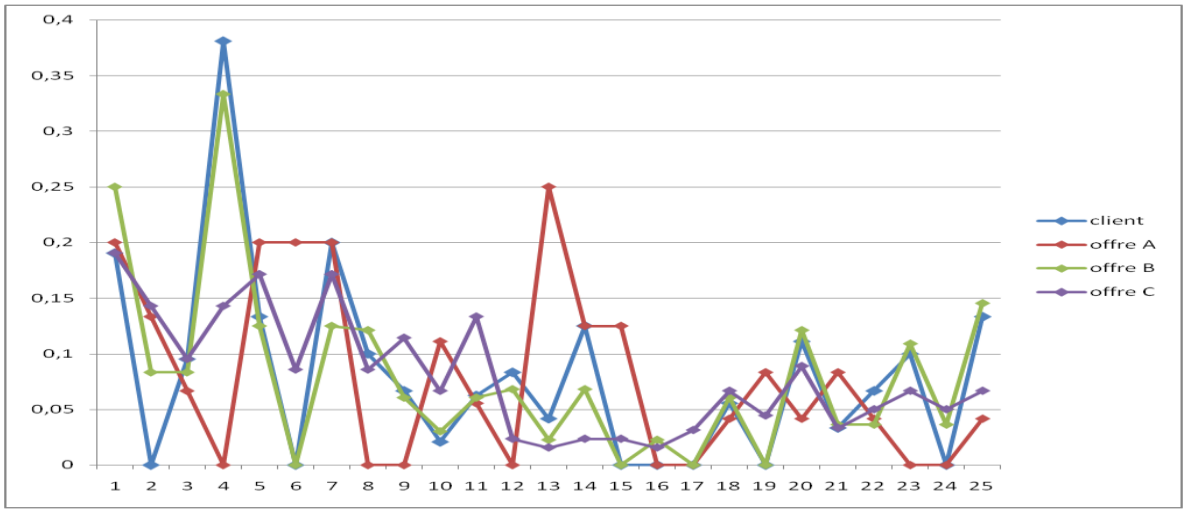


Figure 21. Diagrammes des poids effectifs des différents offres SaaS ainsi que les préférences

La figure 21 montre des diagrammes des poids effectifs des attributs des offres SaaS : A, B et C ainsi que les poids effectifs selon les préférences du client. Nous pouvons remarquer que le diagramme qui se rapproche le plus des préférences du client c'est celui de l'offre B.

	Facteurs	attribut	Offre A $P_{A/pref}$	Offre B $P_{A/pref}$	Offre C $P_{A/pref}$
Fonctionnalité	CRM	Marketing direct	0,03809524	0,04761905	0,03628118
		Ventes CRM	0	0	0
		Centre d'appels	0,00634921	0,00793651	0,00907029
		Aide de dépannage	0	0,12698413	0,05442177
	Collaboration	Communication	0,02666667	0,01666667	0,02285714
		Conférence	0	0	0
		Coordination	0,04	0,025	0,03428571
Préférences	Réputation	Le nombre de clients	0	0,01212121	0,00857143
		Valeur de la marque	0	0,0040404	0,00761905
	Coût	Abonnement annuel	0,00231481	0,00063131	0,00138889
		Coût_implémentation	0,00347222	0,00378788	0,00833333
	Utilisabilité	Fiabilité	0	0,00568182	0,00196078
		Interface ergonomique	0,01041667	0,00094697	0,00065359
		L'attribut d'aide	0,015625	0,00852273	0,00294118
		Sup_dispositif mobile	0	0	0
		Le support « off line »	0	0	0
		Intégration	0	0	0
	Architecture	Multi-client	0,00231481	0,003367	0,0037037
		Scalabilité	0	0	0
		Sécurité	0,00462963	0,01346801	0,00987654
	Configurabilité et personnalisation	Structure de données	0,00277778	0,00121212	0,00111111
		IHM	0,00277778	0,00242424	0,00333333
		Workflow	0	0,01090909	0,00666667
		Struct_organisationnelle	0	0	0
		Contrôle d'accès	0,00555556	0,01939394	0,00888889
		SOMME	0,16099537	0,31071308	0,2219646

Tableau 23. Les poids effectifs des attributs de chaque produit en fonction des préférences du client

Le tableau 23 illustre la troisième partie de notre algorithme qui est le calcul du poids effectif de chaque attribut selon les préférences voulues. En se basant sur la somme de ces poids pour chaque produit, nous pouvons constater que l'offre B est la plus appropriée.

3 LA METHODE DE GENERATION D'UN MODEL DE SOUSCRIPTION AUX SAAS

La souscription est une étape clé qui relie un fournisseur de services à un consommateur. En outre, elle est liée à d'autres activités du cycle de vie telles que le service de package et de la facturation. Dans cette section, nous introduisons un modèle de souscription pour les SaaS. Nous détaillons ensuite, l'approche que nous proposons pour la conception d'un modèle de souscription qui s'occupe de la gestion d'abonnement aux applications SaaS. Pour illustrer le modèle de souscription, des graphiques et des expressions formalisées sont utilisés.

3.1 La souscription pour les SaaS

En règle générale, les SaaS ont un cycle de vie différent des autres logiciels traditionnels. Les étapes telles que l'analyse, le développement et les tests sont fondamentaux. Toutefois, des nouvelles activités sont exigées en plus. La figure suivante montre un cycle de vie global d'une application SaaS.

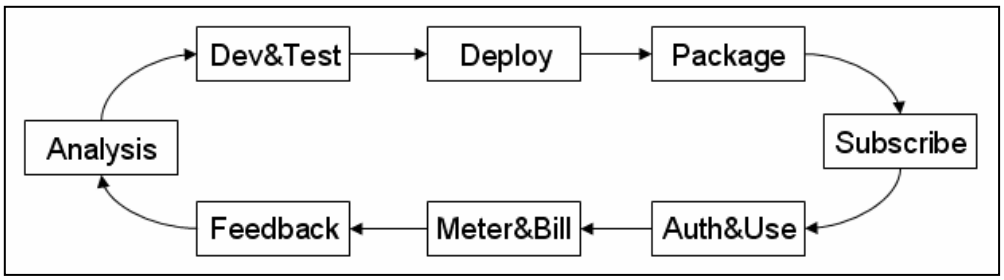


Figure 22. Le cycle de vie d'une application SaaS (Jiang, Sun, Tang, Snowdon, & Zhang, 2009).

Après le développement d'une application et son déploiement, l'étape suivante nommée packaging concerne la définition des termes opérationnels, l'application des politiques de facturation, et ainsi de suite avant qu'elle soit prête à être utilisée en tant que service. Un client peut ainsi souscrire à cette application.

Pour s'inscrire à un service SaaS, il existe plusieurs manières. Si un client s'abonne avec succès à une offre de service, il est donc autorisé à y accéder par la suite. Afin de facturer le nombre et le temps d'accès d'un client à une application SaaS, un modèle de facturation est mis en place en se basant sur des informations mesurant l'utilisation de ces applications.

Habituellement, les deux étapes suivantes : « *feedback* » et analyse « *analysis* » sont essentielles pour l'amélioration d'une application ou d'un service SaaS. Dans la pratique, le cycle de vie peut être différent de celui montré en figure 22.

Dans l'étape de la souscription, un service est instancié en fournissant au client les paramètres spécifiques de la politique de facturation et la configuration de niveau de service. Une autorisation d'accès au service est donnée en se basant sur les résultats de la souscription et en gérant les contraintes ou les règles d'accès. Par exemple, le nombre maximal d'utilisateurs autorisés à accéder au service est spécifié au moment de la souscription comme une propriété de configuration et il devient une contrainte d'autorisation après.

Pour générer une liaison flexible entre l'aspect technique d'une application SaaS (implémentation) et l'aspect métier (business, fonctionnement), nous avons affiné la notion de service, pour les applications SaaS généralement, en un élément de service et une offre de service en prenant comme référence le travail de (Jiang, Sun, Tang, Snowdon, & Zhang, 2009).

La figure ci-dessous montre le modèle de souscription et les entités clés sont décrites comme suit :

- Un élément de service est l'unité d'accès et de mesure. Il peut y avoir des dépendances entre les éléments du service. Un élément de service a éventuellement des types de clients ciblés.
- Un élément de service peut être instancié en une instance de service. Après l'authentification, un client est autorisé à accéder aux instances de service d'une offre. Un élément de service est implémenté par un seul fournisseur.
- L'offre de service est l'unité de souscription comportant un ou de plusieurs éléments de service. Il est destiné à être utilisé par un ou plusieurs types de clients, et il est offert par un seul fournisseur. Des conditions d'utilisation et des politiques de facturation sont définies pour une offre de service.

La souscription permet aux clients de s'inscrire à une offre de service, alors que l'autorisation permet à un utilisateur d'accéder à une instance d'un élément de service.

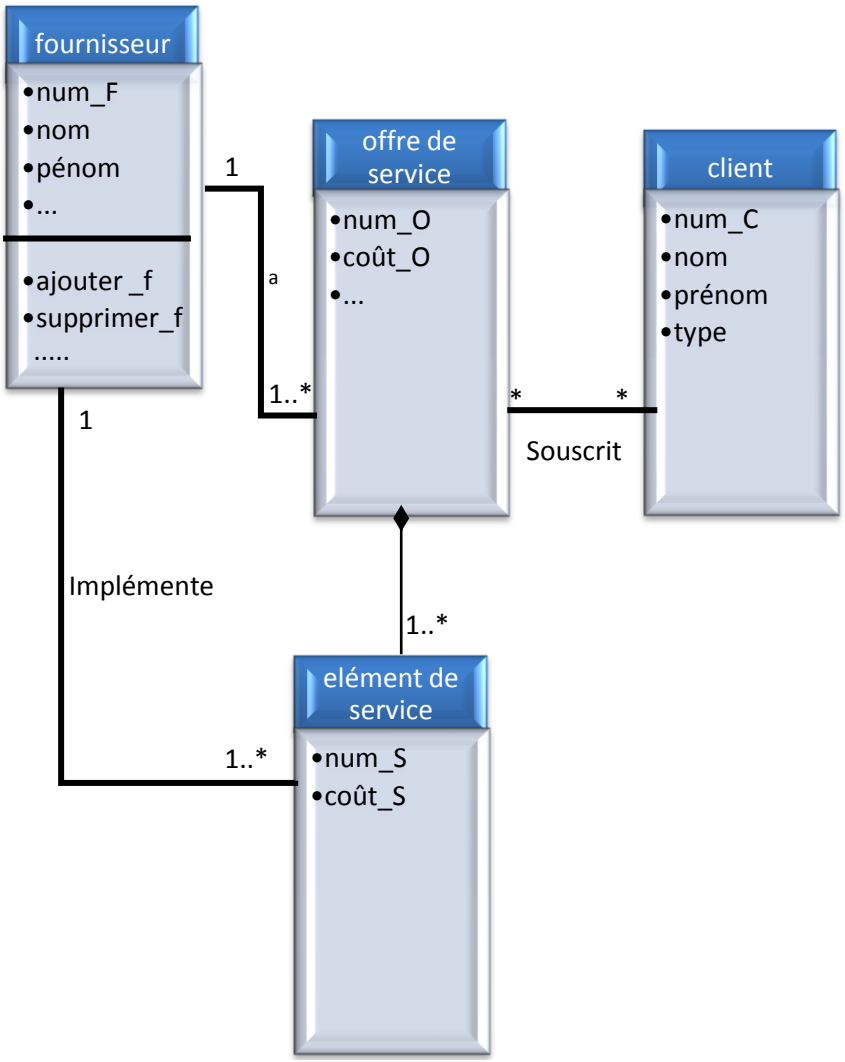


Figure 23. Un diagramme de classes représentant les entités principales d'une application SaaS

Du point de vue écosystème, les rôles suivants peuvent être extraits concernant la gestion de la souscription :

- Le client (ou locataire) est celui qui utilise les services. Il peut y avoir plusieurs types de clients : un particulier ou une société. Donc, un client s'inscrit directement à l'offre pour leur

propre utilisation quand il s'agit d'un particulier, ou bien il s'abonne en tant qu'intermédiaire et permet aux employés d'y accéder dans le cas d'entreprise.

- Le fournisseur est celui qui possède et fournit les services.
- Le client peut abonner avec une offre ou plusieurs offres de service.
- Un fournisseur peut s'inscrire avec une offre de service comme il peut utiliser un élément de service pour le compte de ses clients.

3.1.1 L'Approche à base des patrons

Le modèle de souscription rassemble différents types de considérations autour des entités qui sont les éléments de service et les offres de service, construisant ainsi une base solide pour la gestion d'abonnement aux applications SaaS. Cependant, dans la pratique, les relations entre ces entités et les rôles de l'écosystème peuvent être très complexes.

Basé sur une étude de (Jiang, Sun, Tang, Snowdon, & Zhang, 2009) où ils ont résumé les types de relations en deux groupes de modèles principaux qui sont : la structure de service et les interactions métier. Ces deux groupes de modèles se concentrent sur les connexions entre les entités clés et les interactions entre les rôles de l'écosystème respectivement.

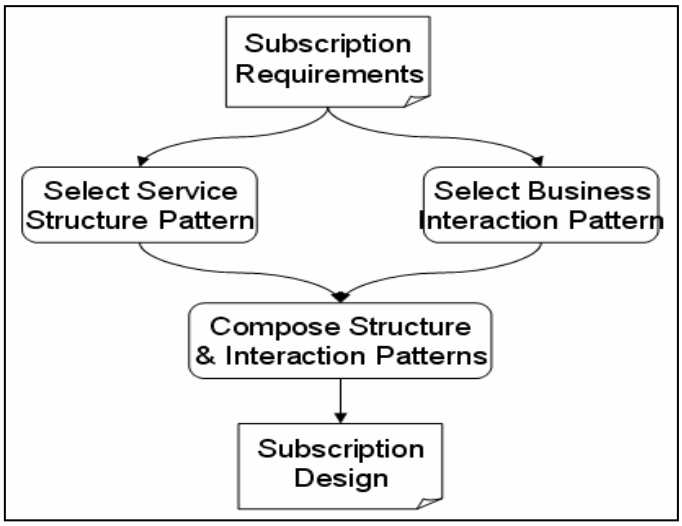


Figure 24. Approche à base de patron

Comme illustré par la figure 24, après que les deux groupes de modèles ont été choisis, ils seront composés pour obtenir la conception finale du modèle de la souscription. Plusieurs modèles peuvent être sélectionnés pour chaque groupe.

3.1.2 Représentation graphique

Pour illustrer la conception de la souscription, une série de représentations graphiques sont inspirées de (Jiang, Sun, Tang, Snowdon, & Zhang, 2009). On se basant principalement sur leur modèle de représentation, on définit graphiquement les entités clés et leurs relations comme le montre le tableau ci-dessous.

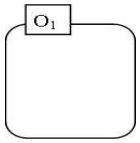
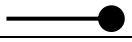
	Représentation graphique
Offre de service	
Elément de service	
Dépend de	
Client	
Fournisseur	
Souscrit	
Possède	

Tableau 24. Représentations graphique pour les entités, les rôles et les relations

Ici « possède » et « souscrit » illustrent une relation entre un rôle et une entité. La relation «contenir» est symbolisée par la disposition directe. Cette figure montre une offre de service O1 qui contient des éléments de service S1 et S2, S2 dépend de S1.

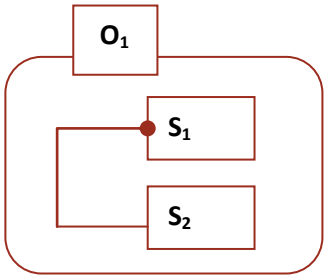


Figure 25. Un Exemple de la Représentation graphique des entités et des relations

L'exemple dans la figure suivante montre que multiples clients peuvent s'abonner à une offre de service O1 auprès d'un fournisseur.

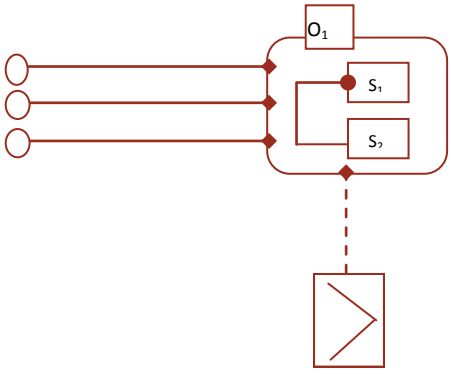


Figure 26. Un exemple de la représentation graphique des rôles et des relations

3.1.3 Expression formalisée

Afin de créer une base solide pour l'automatisation de la souscription et l'analyse, nous avons adopté la formalisation des expressions mises en œuvre par (Jiang, Sun, Tang, Snowdo, & Zhang, A Pattern-based Design Approach for Subscription Management of Software as a Service, 2009). Nous décrivons les entités clés par les formules suivantes :

- Élément de Service : S = (S_ID,{C_type}, F_ID), i = 1, 2, …, n
- Une offre de Service : O = (O_ID, {Si},{C_type},F_ID), i = 1, 2, …, n
- Client: C = (C_ID, C_type, {Oi}), i = 1, 2, …, n
- Fournisseur : F = (F_ID, {Oj}), i = 1, 2, …, n, j = 1, 2, …, m

Où chaque entité est associée à un identifiant (par exemple S_ID, O_ID).

- La relation interne entre les éléments du service est donnée par l'expression suivante où S_m dépend de S_n : $S_m \mid_D S_n$

- L'abonnement à une offre de service est donné par l'expression suivante :

  $Rôle_m \xrightarrow{O_x} Role_n \quad / Role_m \in \{C,F\}, Role_n \in \{C,F\}$

  L'exécution avec succès de la formule ci-dessus, ajoutera O_x à l'ensemble des offres du $Rôle_m$.

- L'ajout d'un élément de service est donné par l'expression suivante :

  $O_i(m) \xrightarrow{S_x} S(n) \quad / m, n \in F$

  L'exécution avec succès de la formule ci-dessus, ajoutera l'élément de service S_x implémenté par le fournisseur n à l'offre O_i du *fournisseur m*.

3.2 Le Modèle de souscription

En se basant sur ce qui existe dans la littérature, nous citons les deux groupes de modèles les plus utilisés pour la souscription des applications SaaS :

- Les patrons de la structure : ce sont ceux qui traitent l'aspect architectural ou structural du service, et

- Les patrons des interactions métiers : ce sont ceux qui représentent l'aspect interaction entre les rôles de l'écosystème et les services.

Dans ce qui suit, nous allons présenter les deux groupes de patrons en détails ainsi que le processus de sélection du patron.

3.2.1 Les Patrons de Structure de Service

Les patrons de la structure de service sont décrits dans le tableau 25. Pour expliquer comment utiliser ces patrons à travers un exemple, nous prenons le cas du modèle de structure des services multiples et dépendants indiqué dans le tableau. Dans une application à base de transactions telle que la gestion des commandes, le reportage est une fonction très utile, elle est construite sur la base de l'historique des transactions et des activités réalisées. Donc, il y a plusieurs services avec des

dépendances entre ces services. Quand cette fonction est livrée comme une offre SaaS, le modèle des Services Multiples et Dépendants précisément décrit cette structure.

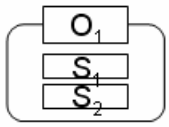
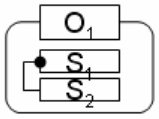
Nom	*Description*	*Expression formalisée*	*Représentation graphique*
Service indépendant	*L'offre du service O_1 contient un ou plusieurs éléments de service. Il y a aucune dépendance entre les éléments*	• *Elément de service : S= {(S_i, {C_type},F_ID)}, i=1,2,…,n* • *L'offre du Service : O= (O_1, {S},{C_type},F_ID)*	
Service Multiple Dépendant	*L'offre du service O_1 contient plusieurs éléments de service. Il y a une dépendance entre les éléments de service. Alors si un élément de service S1 est sélectionné, tous les éléments de service S_i dépendants de ce dernier doivent être sélectionnés.*	• *Elément de service : S=(Si, {C_type}, F_ID) ,i=1,2,…,n.* • *L'offre du Service : O= (O_1, {S},{C_type},F_ID)* • *La Dépendance entre les éléments de service :* $\exists i, j / Si \mid_D Sj$	

Tableau 25. Le patron de structure de service

Le processus de sélection d'un modèle de structure de service est illustré à la figure 27 comme un arbre de décision.

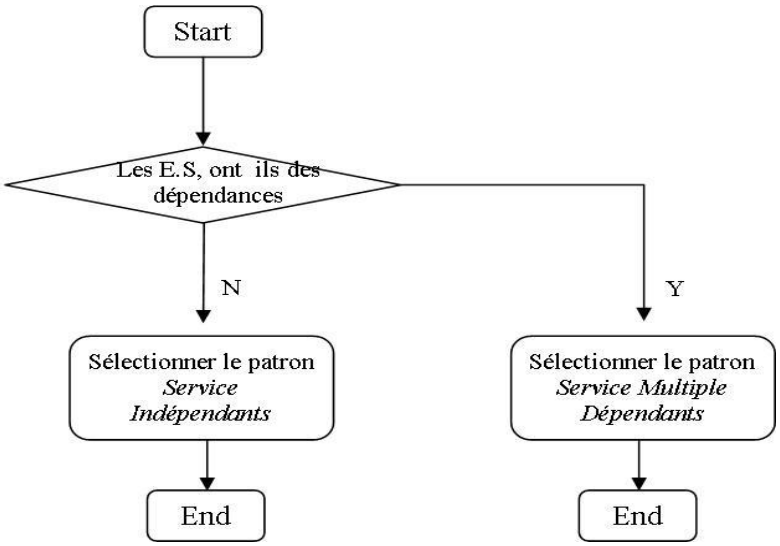


Figure 27. La Sélection d'un Patron de Structure de Service

Pour sélectionner un patron de structure de service, on doit tester s'il y a une dépendance entre les éléments de service d'une offre ou pas. Donc, le patron de Services Indépendant est sélectionné s'il n'y aucune dépendance entre les éléments. Sinon, le patron de Services Multiples Dépendants est sélectionné.

3.2.2 Les patrons des interactions métiers

Les patrons des interactions métiers sont décrits dans le tableau suivant. Les résultats de l'exécution des actions de la souscription sont également définis.

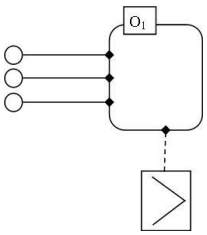
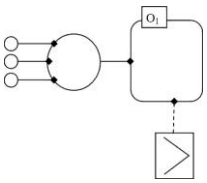
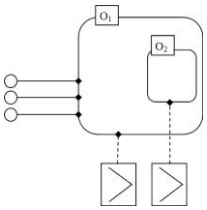
Nom	Description	Expression formalisée	Représentation graphique
Self Service	*Le client s'inscrit directement à l'offre du service d'un Fournisseur.*	• *Elément de service : $S=\{(S_i,\{C\_type_1\}, F\_ID)\}$, $i=1,2,\ldots,n$* • *L'offre du Service : $O=(O_1, \{S\},\{C\_type_1\},F\_ID)$* • *Client : $C=\{C_i, C\_type_1, \{\Phi\}\}$, $i=1,2,\ldots,n$* • *Fournisseur : $F=\{(F_1, O')\}$ Ici $O \in O'$* • *La souscription à l'offre de service :* $Ci \xrightarrow{O_1} F1$, $i=1..n$ *Alors :* $C=\{(Ci, C\_type_1, \{\Phi,O\})\}$, $i=1,2,\ldots,n$.	
Hub-Spoke	*Le client de centre « du Hub » souscrit à l'offre du service du fournisseur directement et « les clients Spoke » souscrivent à l'offre du client du Hub. (Ou le client du Hub souscrit à l'offre pour le compte des clients Spoke.).* *Le Hub est autoriser d'accéder à l'ensemble des éléments de service alors qu'il authorise certains client d'accéder a certains éléments de services.*	• *Elément de service : $S=\{(S_i, \{Hub\},F\_ID), (S_j,\{Spoke_k\},F\_ID)\}$, $i=1,2,\ldots,n$. $j=1,2,\ldots,l$.$k=1,2,\ldots,m$* • *L'offre du Service : $O=\{(O_1, \{S\},\{Hub\},F\_ID)\}$* • *Client : $C=\{(hub, Hub, \Phi), (spoke_i, Spoke, \beta)\}$, $j=1,2,\ldots,m$* • *Fournisseur : $F=\{(F_1, O')\}$ Ici $O_1 \in O'$* • *La souscription de l'offre de service :* $hub \xrightarrow{O.S_i} F_1$ $spoke_i \xrightarrow{O.S_j} hub$, $i=1,2,\ldots n$ *Alors:* $C=\{(hub,Hub, \{\Phi ,O.S_i\}), (spoke_k, Spoke, \{\beta, O.S_j\})\}$, $i=1,2,\ldots,n.$ $j=1,2,\ldots,l.$ $k=1,2,\ldots,m.$	
Délégué	*Le fournisseur souscrit à une offre d'un autre fournisseur pour le compte de ses clients. Après la souscription, le fournisseur est autorisé à vendre l'offre de service correspondant.*	• *Elément de service : $S=\{(S_i,\{C\_type\},F_j)\}$ $i=1,2,\ldots,n$ $j=1\ldots m$* • *L'offre de Service : $O=(O_1, \{S\},\{C\_type\},F_x)$* • *Client : $C=\{C_i, C\_type, \Phi\}$, $i=1,2,\ldots,n$* • *Fournisseur : $F=\{(F_x, O'), (F_b, O'')\}$ $b=1,2,\ldots,n$. Ici $O \in O'$et $F_x \neq F_b$* • *La souscription de l'offre de service :* $O(F_x) \xrightarrow{S_y} S(F_b)$ $y=1..n$ $C_i \xrightarrow{O} F_x$ $i=1\ldots n$ *Finalement* $S=\{(S_i,\{C\_type\},F_x), (S_k,\{C\_type\},F_b)\}$ $i=1,2,\ldots,n$ $k=1\ldots m$ $b=1,2,3\ldots l$ *Et* $C=\{(Ci, C\_type, \{\Phi, O\})\}$, $i=1,2,\ldots,n$	

Tableau 26. Les patrons d'interaction Métiers

A travers la figure 28, nous décrivons la sélection des patrons d'interaction métier. Il s'agit d'un arbre de décision dégénérée où tous les modèles peuvent être sélectionnés tant que les conditions proposées peuvent être satisfaites.

.

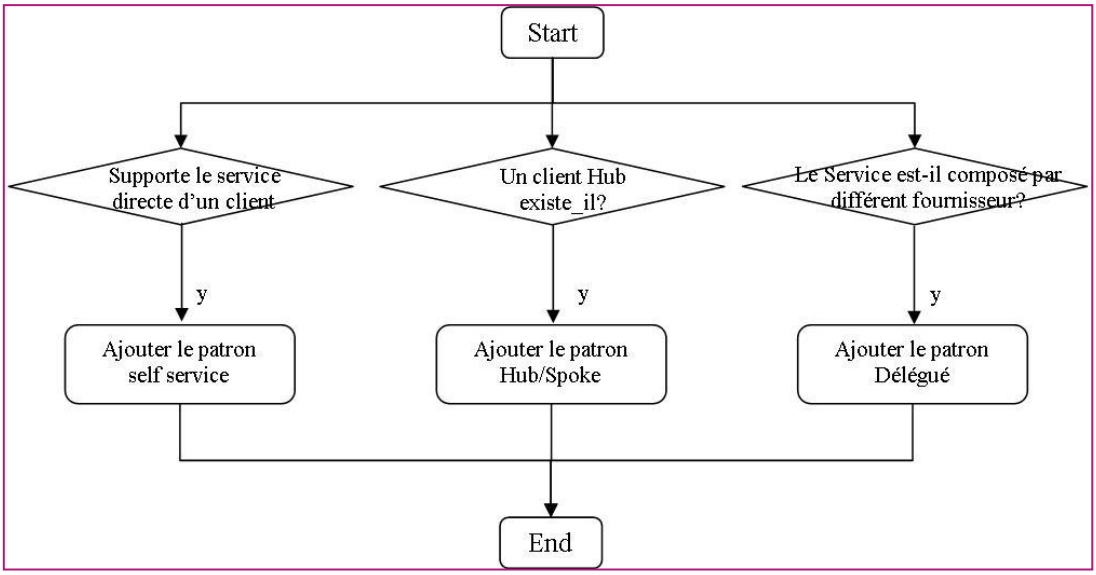


Figure 28. La sélection des patrons d'interaction business

L'accès à l'offre de service peut être direct ou à travers un intermédiaire. Le patron sélectionner doit donc être le self-service ou le Hub/spoke respectivement. Quand les éléments de service sont implémentés par différents fournisseurs, le patron délégué doit être sélectionné aussi.

3.3 Cas d'étude

Dans cette section, nous appliquons l'approche à base de modèle de conception pour la gestion de la souscription sur un cas d'étude. Pour cela, nous avons pris le même exemple de la section (2.4). Donc notre offre est une application CRM avec des modules de collaboration. Dans ce qui suit, nous essayons de choisir un modèle de souscription à l'offre B (voir la section 2.4). Ainsi, nous avons principalement six modules fonctionnels qui sont : Marketing direct(M), Ventes CRM(V), Centre d'appels(CA), Aide de dépannage(AD), Communication(Com) et Coordination (C).

Nous devons tout d'abord choisir la structure de service. Ainsi, en sélectionnant le patron « Services Multiples et Dépendants », la structure suivante est obtenu :

Elément de Service: S = {(M, {C_type_i}), (V, { C_type_i }), (CA, { C_type_i }),(AD, { C_type_i }),(Com, { C_type_i }), (C, { C_type_i })}
Offre de Service : O = (Offre B, {M, V, CA, AD, Com, C})
Les dépendances des éléments de Service sont :
$V |_D M, CA |_D M, CA |_D V, CA |_D AD$, AD $|_D V$,$Com |_D M, M |_D Com$, $Com |_D V$, $V |_D Com$, $Com |_D CA$, $CA |_D Com$, $Com |_D AD, AD |_D Com, C|_D M, M |_D C$, $C |_D V$, $V |_D C$, $C |_D CA, CA |_D C$, $C |_D AD, AD |_D C$.

La représentation graphique est illustrée par la figure ci-dessous.

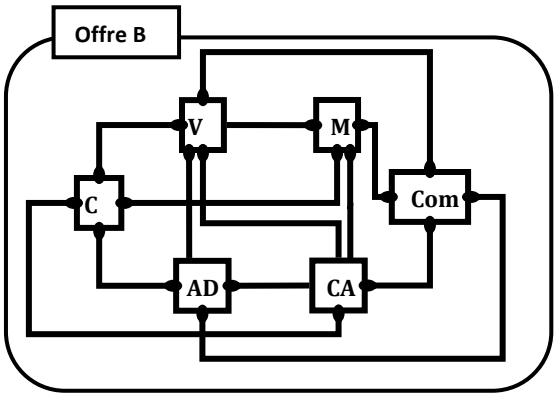


Figure 29. La Structure de Service pour l'offre B

Nous supposons que les modules de collaboration sont implémentés par un autre fournisseur donc le patron d'interaction business « délégué » doit être choisis. D'une autre part, l'offre B a un niveau de configurabilité et de personnalisation très fort avec un module de contrôle d'accès à niveau très fort aussi. Cela signifie que le patron « Hub-Spoke » doit être choisi aussi, comme illustré par la figure ci-dessous.

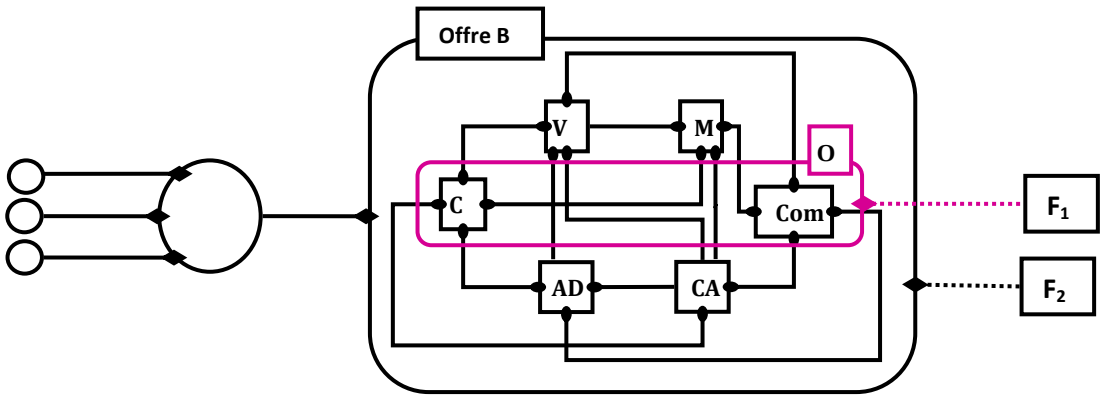


Figure 30. Interaction Business pour l'offre B

L'expression formelle est la suivante :

- *Elément de service : S={ ({M, V, AD, CA}, {C_type},F_2), ({Com,C},{C_type},F_1) }, i=1,2,…,n. j=1,2,…,l.k=1,2,… ,m*
- *L'offre du Service : O= {(Offre B, {S},{ Hub},F_2)}*
- *Client : C= {(hub, Hub, Φ), (spoke$_i$, Spoke, β)}, j=1,2,…,m*
- *Fournisseur : F= {(F_2, Offre B), (F_1, O}*
- *La souscription de l'offre de service :*

$O(F_2) \xrightarrow{S_y} S(F_1)$ $y=1..n$

$hub \xrightarrow{O.S_i} F_2$

$spoke_l \xrightarrow{O.S_j} hub$, $i=1,2,…n$

Alors:

C= {(hub,Hub, {Φ ,Offre B.S_i}), (spoke$_k$, Spoke, { β, Offre B.S_j})}, i=1,2,…,n. j=1,2,…,l. k=1,2,… ,m.

Ainsi, en appliquant de la méthode de souscription sur l'offre B (voir section 2.4), nous avons pu choisir la composition entre les patrons de service multiple dépendant et délégué. Nous tenons à mentionner que d'autres compositions peuvent être faites en changeant les suppositions. Par exemple, si nous supposons cette fois que touts les modules fonctionnelles sont implémenté par le même fournisseur, le patron délégué ne peut pas être choisi et ainsi de suite.

4 CONCLUSION

Nous avons développé dans ce chapitre deux méthodes.

La première méthode concerne la sélection d'un produit SaaS le mieux adapté aux besoins des utilisateurs. Nous avons constaté que le choix d'un meilleur produit SaaS satisfaisant la plupart des conditions des alternatives disponibles est un problème MCDM où on a besoin de comprendre l'arrangement des conditions et des offres de produit. Le processus de sélection implique des critères et des produits multiples ; par conséquent, la sélection basée sur les jugements n'identifie pas le choix le plus approprié. Le processus de la hiérarchie exige une étape cruciale qui consiste à donner des priorités aux paramètres et aux produits. Cette étape est habituellement exécutée manuellement ou basée sur quelques échelles de jugement qui manquent de rigueur. Notre travail suggère donc l'utilisation d'AHP comme technique quantitative pour aborder ce sujet. L'AHP est employé pour définir la hiérarchie entre les différents facteurs et attributs. En se basant sur cette hiérarchie, nous avons développé une méthode pour calculer les poids des différents attributs des paramètres de choix d'une part et le score des produits d'une autre part. Ceci pour avoir un choix plus au moins raisonnable qu'une simple opinion subjectif.

Pour la deuxième méthode, nous avons remarqué que la complexité des applications SaaS et le modèle métier associé ont introduit des défis pour la gestion de la souscription. La gestion de souscription des services web existante n'aborde pas ces défis. Nous avons modifié la méthode présentée par [Jiang et al., 2009] pour guider un fournisseur d'application SaaS dans la conception du module de gestion de la souscription selon l'application, ses caractéristiques et le modèle métier associé. Pour cela, il a fallu analyser les applications SaaS afin de définir le patron de la structure et celui des interactions. Ces deux derniers sont les éléments clés de cette méthode. Ces éléments ont été conclus après analyse de divers types d'entreprises de SaaS.

SIXIEME CHAPITRE : L'ARCHITECTURE DETAILLEE DU SERVICE BROKER (MEDIATEUR)

1 INTRODUCTION

Dans le quatrième chapitre, nous avons vu que beaucoup de travaux de recherche ont évoqué la nécessité d'un service broker ou médiateur pour les applications SaaS. Le but principal derrière la conception d'une couche intermédiaire pour les applications SaaS est la mise en œuvre d'un environnement de confiance pour faciliter la communication et l'échange de données entre clients et fournisseurs. D'une autre part, le médiateur permet l'automatisation de la recherche et la sélection d'une application SaaS qui répond le mieux aux besoins des clients. Elle permet aussi aux fournisseurs d'applications, d'exposer leurs services dans cet environnement de confiance et de les définir avec le plus de précision possible pour une utilisation optimale.

Dans le monde réel, l'agent immobilier est un bon exemple car ses processus et fonctions imitent parfaitement le service broker SB. Le but de ce dernier consiste à gérer les requêtes et les réponses. Il agit comme un agent double. Il effectue des tâches de performance et assure la confiance pour les fournisseurs d'une part, et d'autre part, il s'assure que les services demandés correspondent le mieux à ce qu'il recherchait. Les interactions entre les entités de l'écosystème et le SB peuvent être présentées comme suit :

- Client / Broker : le Broker gère les requêtes des clients pour l'enregistrement, la demande des services basés sur certains critères et l'abonnement pour le meilleur service présenté. Il effectue également autres tâches spécifiques pour les clients des applications SaaS telles que le maintien d'un classement des services présentés en fonction de leurs exigences.
- Fournisseur / Broker : les fournisseurs doivent inscrire leurs services au près du Broker qui, à son tour, annonce ces services aux abonnés potentiels. Il leur offre également la possibilité de gérer la souscription et de contrôler l'utilisation du service.
- Client / Fournisseur : en utilisant le Broker, les clients recherchent des services qui correspondent à leurs besoins. Après la recherche dans le serveur UDDI interne, le Broker présente tous les services qui correspondent aux exigences des clients fondées sur certains critères de classement que nous avons déjà vu dans le chapitre quatre. Le Broker doit également assurer la négociation des agréments du niveau de service (SLA) entre les deux parties, surveiller automatiquement la disponibilité du service, surveiller le rendement des entreprises…

L'objectif essentiel de ce travail est de définir une architecture détaillée du service Broker et de réaliser et de valider un prototype de l'approche proposée. Pour cela, nous avons développé notre prototype dans un environnement Windows vista, en utilisant Java comme langage de programmation (JDK 1.6) avec l'environnement de développement NetBeans 7.0. Pour la publication de ce dernier, nous avons utilisé le serveur d'application *Apache* tomcat 5.5.29, *Axis 1.4 et JUDDI*[22] *0.9rc4.*

Étant donné que la principale technologie derrière le service Broker proposé est le Services Web, nous présentons, dans ce qui suit, le service web avec ses différents standards. Nous détaillons par la suite l'architecture du Service Broker avec ses différentes composantes

2 LES SERVICES WEB

Actuellement, les applications SaaS communiquent principalement par le biais de services web. Les services web fournissent un moyen standard d'interopérabilité entre les différentes applications

22 jUUDI is an Open source UDDI server which have to be integrated with the apache tomcat application server. jUDDI provides a console where the uses can publish and inquire businesses / Services and tmodels along with the binding templates. Apache provides the HOW-TO setup guide it its wiki, yet its bit confusing and out of date.

s'exécutant sur une variété de plates-formes. Le concept de services web est également très applicable pour le transfert de données entre les applications ou entre les bases de données. Il n'y a aucune norme réelle pour la communication entre les applications SaaS mais en raison des services web, nous avons une bonne base pour commencer à développer certaines normes de communication. Les trois technologies clés des services Web sont : SOAP, WSDL et UDDI que nous décrirons brièvement dans ce qui suit :

1. **SOAP « Simple Object Access Protocol » :** Proposé au W3C par Microsoft, IBM, Lotus, DevelopMentor et Userland, SOAP est un protocole de communication entre services web. Il fournit les mêmes fonctionnalités que les technologies CORBA, DCOM et Java RMI, mais comme il est basé entièrement sur XML cela le rend indépendant des langages et des plateformes d'exploitation employées pour l'implémentation des services web (Maesano, Bernard, & Le Galles, 2003). Par exemple, un client SOAP Java s'exécutant sur Linux ou un client SOAP Perl s'exécutant sur Solaris peut se connecter à un serveur SOAP Microsoft s'exécutant sur Windows 2000 (Chelbabi, 2006).

2. **WSDL « Web Service Description Language » :** Développé par Microsoft, Ariba et IBM (Smith, Daconta, & Obrst, 2003), le WSDL est une application XML créé pour décrire et publier les interfaces et protocoles des services Web d'une manière standard. L'interface d'un service web est nécessaire, car elle permet d'éviter de décrire des interactions spécifiques pour chaque serveur web. WSDL présente un format commun pour la description et la publication des interfaces et protocoles relatifs aux services web. Une description WSDL d'un service web est faite sur deux niveaux ; un niveau abstrait et un niveau concret.

3. **UDDI (Universal Description Discovery and Integration) :** Introduit en 2000 par Ariba, Microsft et IBM, UDDI (Newcomer, 2002) a été créé pour faciliter la découverte de services web en plus de leurs publications. UDDI est un annuaire orienté Business pour services web. Les organisations publient les informations décrivant leurs services web dans l'annuaire. L'application client, ayant besoin d'un certain service, consulte cet annuaire pour la recherche des informations concernant le service web fournissant le service désiré pour une éventuelle interaction (Chelbabi, 2006) (Maesano, Bernard, & Le Galles, 2003). UDDI est subdivisé en deux parties principales :

    - La partie publication qui regroupe l'ensemble des informations relatives aux entreprises et à leurs services. Ces informations sont introduites via une API d'enregistrement.
    - La partie découverte qui facilite la recherche d'information contenue dans UDDI grâce à l'API SOAP.

3 L'ARCHITECTURE DU BROKER

Une fois le fournisseur se connecte à notre service broker, il se voit afficher des interfaces lui permettant la modification de n'importe quel service qu'il a déjà sur le système ou l'ajouter un nouveau service. En cas d'ajout, le fichier WSDL de ce service doit être étendu pour ajouter des attributs d'interface. Lorsque cette option est terminée, une copie du nouveau fichier WSDL conforme et étendue devrait être renvoyée au fournisseur afin qu'il puisse le localiser pour un accès ultérieur.

Lorsqu'un client souhaite s'abonner à un nouveau service, il utilise le broker pour rechercher les services disponibles. Une fois le client identifie un service, le broker peut consulter le profil de l'utilisateur afin de recueillir la plupart des informations pour l'inscription. Toute information

nouvelle ou spécifique est demandée de l'abonné par le broker et elle est stockée plus tard dans son profil.

Le broker analyse le fichier WSDL et le présente à l'abonné en forme d'interface générique. Une fois l'utilisateur est souscrit à un nouveau service, ce dernier sera mis sur son Bureau ou sur son écran principal.

La figure 31 illustre une architecture conceptuelle du service Broker. Nous décrivons la fonctionnalité des diverses composantes et par la suite les interactions entre elles.

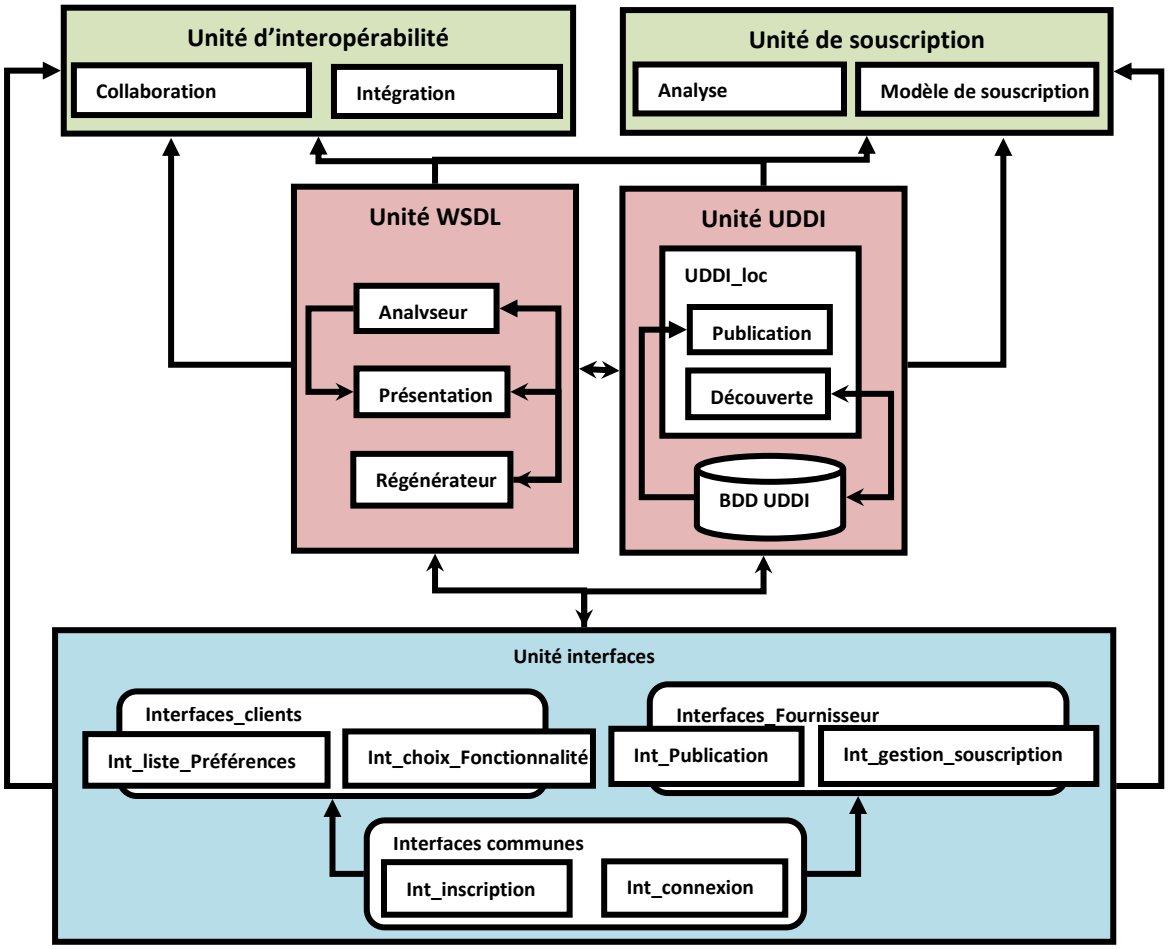


Figure 31. L'architecture du Broker

Notre service broker est composé de cinq unités principales qui sont :

- L'unité interface : interagie directement avec les fournisseurs et les clients des applications SaaS
- L'unité UDDI : gère la publication et la découverte d'une application SaaS
- L'unité WSDL : responsable de la gestion des fichiers WSDL.
- L'unité de souscription : aide les fournisseurs de service à choisir un modèle de souscription.
- L'unité d'interopérabilité : responsable de la collaboration entre les différents fournisseurs des applications SaaS et l'intégration de ces derniers dans les systèmes existants.

Chaque unité peut être décomposée en plusieurs éléments. Un élément peut avoir plusieurs composants. Dans ce qui suit, nous détaillons les différentes unités de notre médiateur.

3.1 Les interfaces utilisateur

C'est l'interface graphique (GUI) que les utilisateurs utilisent pour accéder au Broker. L'implémentation de cette interface est typiquement basée sur un navigateur internet. Il s'agit d'un client léger. Nous regroupons les interfaces sous trois sous-ensembles à savoir :

1. **Des interfaces communes** qui représentent la section où un utilisateur donné veut se connecter à notre service broker. Pour cela, il doit avoir un pseudonyme et un mot de passe. S'il s'agit d'un nouvel utilisateur, il doit s'inscrire pour pouvoir accéder aux autres fonctionnalités du SB.

2. **Des interfaces clients** qui représentent la section où les clients de services interagissent avec le Broker. Elles sont dédiées principalement à la découverte d'une application SaaS en fonctions des besoins. Les critères de choix sont regroupés dans deux sous ensembles à savoir : les fonctionnalités recherchées et la liste des préférences (optionnelles). Pour cela, deux interfaces clients ont été définis :

    - int_choix_Fonctionnalité : pour préciser les fonctionnalités recherchées d'une part et le type de compte recherché «unique ou multiple » d'autre part.

    - Int_liste_Préférences : pour préciser la liste des préférences et le nivaux de compétence recherché pour chaque paramètre.

3. **Des interfaces fournisseur de services «Providers Interface »** qui représentent la section où les fournisseurs de services interagissent avec le Broker. Elles sont dédiées principalement à la publication de toutes les informations techniques d'une application SaaS, la gestion de la collaboration, et la gestion et la génération du modèle de souscription. Pour cela, trois interfaces fournisseurs ont été définis :

    - Int_Publication : pour la saisie de l'URL du WSDL de l'application SaaS et tous les paramètres et les niveaux de compétences des différents attributs définis au cinquième chapitre (voir section 2.1 et 2.2).

    - Int_gestion_souscription : pour définir tous les éléments de services ainsi que la structure de l'offre et les interactions avec l'écosystème « interactions métiers » (voir section 3.2).

3.2 L'unité UDDI

Elle se compose de deux éléments détaillés ci-après et qui sont : Le register UDDI local « Local UDDI Registry»(UDDI_loc) et la base de données UDDI.

3.2.1 Le register UDDI local « Local UDDI Registry»

Il autorise et accepte l'enregistrement des applications SaaS autonomes. Les fournisseurs s'inscrivent directement au UDDI local en remplissant un formulaire web mis en disposition par le SB. Le Registre local vérifie que l'enregistrement est fait correctement. Après la vérification, il enregistre les informations d'inscription dans la BDD_UDDI « Base de données UDDI » et les récupère lors de la demande. Nous utilisons les structures de données UDDI pour :

- Définir les conditions d'accès et d'utilisation et la relation entre les fournisseurs. La structure utilisée est « publisherAssertion ».

- Faire une spécification technique et surtout ajouter des attributs pour une meilleure sélection des services les plus appropriés aux besoins des utilisateurs. La structure utilisée est « tModel ».

Notre UDDI_loc a principalement deux composantes, à savoir :

- **La composante de publication** : qui regroupe les fonctions de navigation, de recherche et de consultation des informations de l'annuaire ;
- **La composante de découverte** : qui regroupe les fonctions de publication, de création et de modification ou de suppression des informations de l'annuaire.

3.2.1.1 La publication

Lorsque l'administrateur d'une organisation (entreprise, administration, association … etc.) souhaite publier des informations dans UDDI_loc, il doit interagir avec l'annuaire par le biais d'une interface dédiée aux fonctions de publication. Cette composante se base principalement sur l'API de publication qui comporte seize fonctions organisées en quatre groupes homogènes comme le montre le schéma suivant :

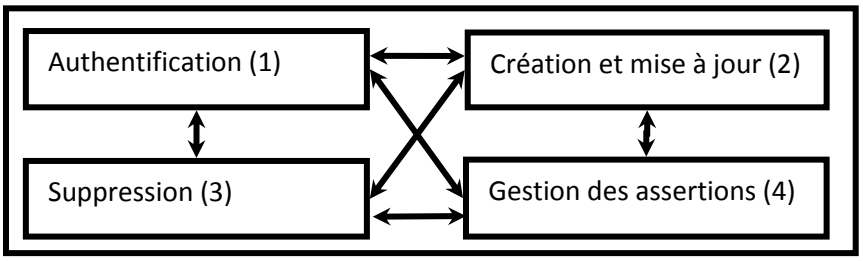


Figure 32. Les API de publication

Le premier groupe comporte les fonctions d'authentification :

- La fonction get_authToken demande un jeton d'authentification à l'opérateur de l'annuaire. Ce jeton est requis pour toutes les autres fonctions de l'API de publication ;
- La fonction discard_authToken demande l'invalidation du jeton préalablement fourni par la fonction get_authToken ;
- La fonction get_registeredInfo demande un résumé de l'information gérée par un administrateur UDDI_loc (liste des entités métier et des services types administrés par cette personne).

Le deuxième groupe rassemble les fonctions de création et de mise à jour des structures de données UDDI :

- La fonction save_business enregistre une nouvelle entité métier ou met à jour une entité métier existante ;
- La fonction save_service enregistre un nouveau service métier ou met à jour un service métier spécifique à l'intérieur d'une entité métier ;
- La fonction save_tModel enregistre un nouveau service type ou met à jour un service type existant.

Le troisième groupe est constitué des fonctions de suppression des structures de données UDDI_loc :

- La fonction delete_business supprime des informations enregistrées pour une entité métier ;

- La fonction delete_service supprime un service spécifique à l'intérieur d'une entité métier ;
- La fonction delete_tmodel supprime un service type spécifique. Cette suppression est seulement logique car le service type peut être référencé par ailleurs.

Le quatrième groupe est représenté par les fonctions de gestion des assertions « Relations entre entités métier » :

- La fonction get_publisherassertions recherche des assertions relationnelles associées au compte de l'utilisateur ;
- La fonction add_publisherassertions ajoute des assertions relationnelles à l'ensemble des assertions existantes, associées au compte de l'utilisateur ;
- La fonction set_publisherassertions affecte des assertions relationnelles, associées au compte de l'utilisateur ;
- La fonction delete_publisherassertions supprime des assertions relationnelles à un ensemble d'assertions existantes, associées au compte de l'utilisateur ;
- La fonction get_assertionstatusreport demande un rapport sur la situation des assertions relationnelles associées au compte de l'utilisateur.

3.2.1.2 La découverte

La composante de la découverte comporte huit fonctions :

1. La fonction find_business : cette fonction recherche l'existence d'information sur une ou plusieurs entités métier. Elle renvoie un message businessList.
2. La fonction find_service : cette fonction recherche l'existence de services métier spécifiques à l'intérieur d'une entité métier. Elle renvoie un message serviceList.
3. La fonction find_tModel : cette fonction recherche l'existence d'un ou plusieurs services types. Elle renvoie un message tModelList.
4. La fonction get_businessDetail : cette fonction recherche une information complète sur une ou plusieurs entités métier. Elle renvoie un message businessDetail.
5. La fonction get_businessDetailExt : cette fonction recherche une information étendue sur une ou plusieurs entités métier. Elle renvoie un message businessDetailExt.
6. La fonction get_serviceDetail : cette fonction recherche une information complète sur un service métier spécifique à l'intérieur d'une entité métier. Elle renvoie un message serviceDetail.
7. La fonction get_tModelDetail : cette fonction recherche une information complète sur un service type. Elle renvoie un message tModelDetail.
8. La fonction find_relatedBusinesses : cette fonction recherche des entités métier associées à une entité métier donnée. Elle renvoie un message relatedBusinessesList.

3.2.2 La base de données UDDI « The UDDI Database »

La base de données UDDI est utilisée pour stocker des informations de toutes les applications inscrites. Pour implémenter et manipuler des données dans cette base, nous avons utilisé juddi avec sa version 0.9rc4, le MySQL JDBC « Java Database Connectivity » Driver avec sa version mysql-connector-java-5.0.4 et le MySQL Server avec sa version 5.0.26.

La figure ci-dessous montre un diagramme simplifié des différentes classes de la base de données UDDI.

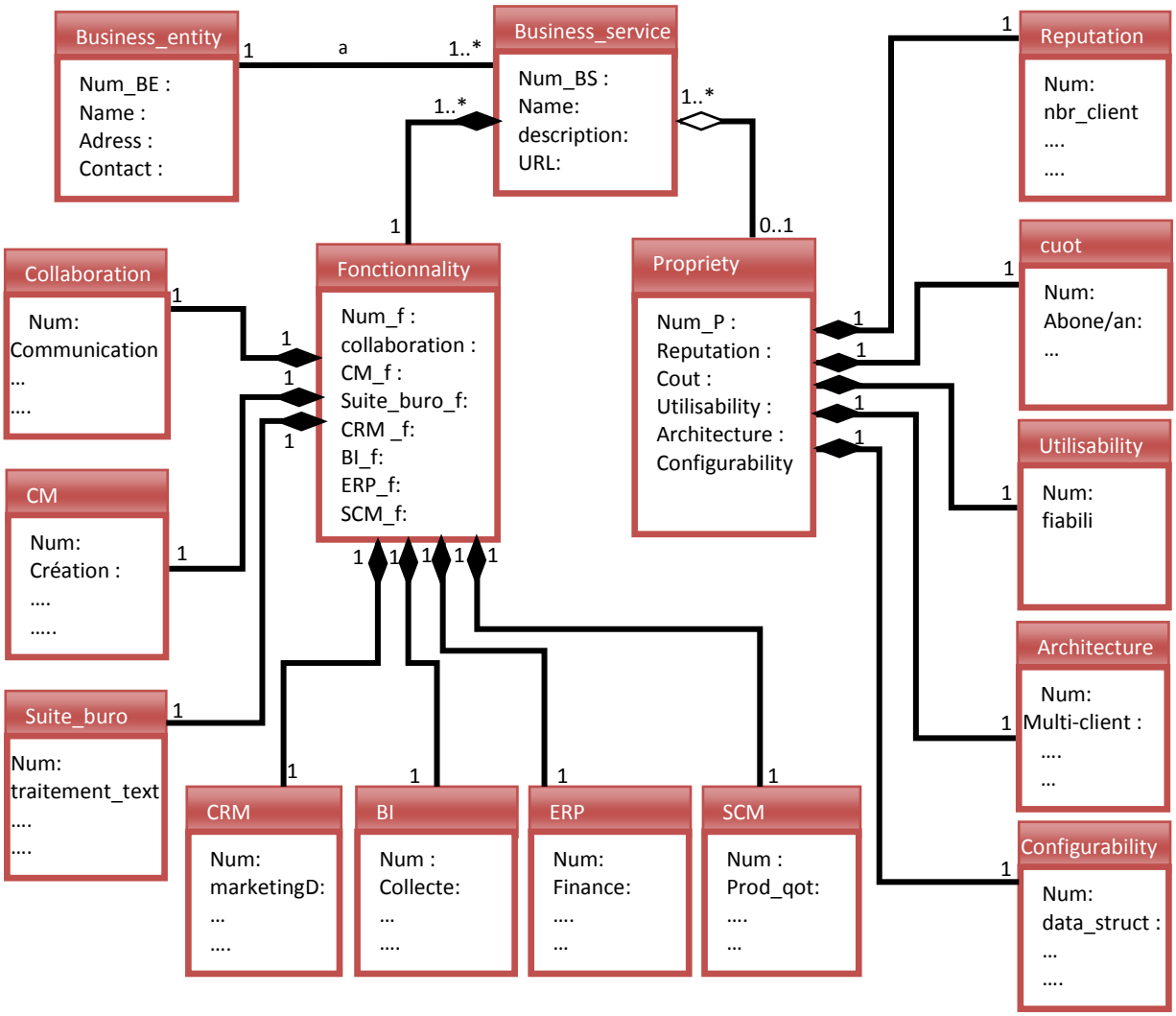


Figure 33. diagramme de classes de la base de données UDDI

- *Business_entity* : représente les fournisseurs qui possèdent et fournissent les services. Chaque fournisseur peut offrir un ou plusieurs services.
- *Business_service* : représente les offres de service. Chaque *Business_service* est offert par un seul fournisseur, il doit avoir des fonctionnalités et il peut avoir d'autres propriétés.
- *Fonctionnality :* représente les nivaux de compétence des différents facteurs de fonctionnalité déterminés dans le chapitre précédent (voir section 2.1). les différents facteurs sont aussi représentés par des classes comme montre la figure 33. Ces derniers définissent les nivaux de compétences des différent attributs (voir chapitre 5 section 2.1).
- *Propriety :* une offre de services peut avoir d'autres propriétés telles que la réputation, le cout, l'utilisabilité, l'architecture et la configurabilité (voir chapitre 5 section 2.2).
- Un fournisseur peut s'inscrire avec une offre de service comme il peut utiliser un élément de service pour le compte de ses clients.

3.3 Unité d'interopérabilité et d'intégration

Le Broker que nous proposons exige aux fournisseurs d'applications SaaS d'exposer des services prédéfinis pour les divers besoins de communication et ainsi l'interopérabilité.

La communication et l'intégration de données sont une responsabilité énorme pour les entreprises qui veulent investir dans des applications SaaS. L'important est de savoir comment échanger des informations entre les différents services et comment assurer que les données sont authentifiées et n'ont pas été endommagées durant la transition. Pour cela, il faut étendre le fichier WSDL pour mieux définir les valeurs retournées. Le fournisseur doit avoir l'opportunité d'étendre le WSDL de leur propre grâce en utilisant un formulaire web fourni par le Broker.

Notre gestionnaire d'interopérabilité contient deux composantes :

- Une composante de collaboration : qui utilise la notion du WSDL étendu pour bien définir toutes les conditions d'utilisation entre les différents fournisseurs. Elle exploite aussi la structure de données d'UDDI « *publisherAssertion* » pour définir les relations entre les entités métier.

- Une composante d'intégration : responsable sur l'intégration des données échangées entre applications SaaS. Cette composante utilise les services web comme un mécanisme standard. Par exemple, l'utilisation de SOAP comme une enveloppe de message commune pour les communications entre les plates-formes SaaS et leurs clients.

3.4 Le gestionnaire du document WSDL « WSDL Manager » (WSDL_Mg)

Cette composante s'occupe de la gestion des opérations sur les fichiers WSDL. Le Broker nécessite une extension de fichier WSDL du fournisseur dans deux situations :

- Tout d'abord, le fichier WSDL est étendu afin d'ajouter des attributs d'interface à un service. Cela signifie que désormais une interface de service peut avoir des caractéristiques pour définir correctement ses fonctionnalités, utilisation, coût, disponibilité, fiabilité ... etc.

- Dans le deuxième cas, une extension du fichier WSDL est requise pour la communication des données et l'interopérabilité entre les différents fournisseurs des applications SaaS.

Dans le WSDL_Mg, il y a trois sous-composantes. Chacune d'elles effectue une tâche spécifique.

1. **L'analyseur :** qui prend en entré n'importe quel fichier WSDL et il analyse ses paramètres d'E/S ainsi que les informations d'exécution. L'analyseur également extrait les informations WSDL tel qu'elles sont définies par le Broker.

2. **La composante de la présentation** : est responsable sur toutes les présentations de WSDL qui doivent être fait. Il existe deux situations où le fichier WSDL doit être présenté.
   - La première est lorsque le client veut tester un service avant de souscrire. Le présentateur prend le fichier WSDL analysé de l'analyseur et le convertit en un formulaire web en établissant tous les paramètres d'entrée requis et les attributs d'interface supplémentaire incluses dans le fichier. Cela permet au client de tester le service en remplissant les paramètres d'entrée et d'exécuter le service après.
   - La deuxième est lorsque la présentation ajoute des champs supplémentaires au formulaire web pour que les fournisseurs l'utilisent afin de définir des types de données complexes ou des informations sur des paramètres supplémentaires.

3. **Le régénérateur** : est la partie de la WM qui reconstruit le fichier WSDL pour le fournisseur après que les informations supplémentaires sont ajoutées. Le nouveau fichier WSDL est ensuite envoyé vers le fournisseur. Ce dernier remplace l'ancien WSDL par le nouveau fichier WSDL étendu. Ce composant garantit que le fichier WSDL soit construit d'une manière appropriée et qui adhère avec le schéma WSDL du Broker afin qu'il puisse être lu et utilisé par le Broker.

3.5 La gestion de la souscription

Comme le SaaS adopte un modèle de tarification en fonction de l'utilisation, le client doit être souscrit d'abord à un service afin de l'utiliser. Le fournisseur de services SaaS doit définir tous les services disponibles et comment s'abonner à chacun d'eux. De plus, le nombre de fournisseurs qui commencent à explorer le modèle SaaS est devenu de plus en plus important. Ces derniers introduisent différents types d'applications et des modèles métier [Ma et al., 2008].

Sachant qu' :

- Une application complexe peut avoir différents types de locataires.
- Une application peut être décomposée en plusieurs éléments de service pour offrir plus de flexibilité aux clients.
- Un client peut souscrire directement à une offre pour son propre compte, comme il peut souscrire pour le compte d'autres utilisateurs

Tous ces aspects doivent être gérés par un système de gestion efficace d'abonnement. Nous définissons deux composants pour la gestion de souscription :

- Composant d'analyse : la méthode d'analyse d'applications est basée sur deux types ; la structure de service et les interactions entre les rôles de l'écosystème. L'analyse de la structure du service est utilisée pour la définition des différents éléments d'un service et les différentes offres d'un fournisseur ainsi que les interactions entre ces éléments. L'analyse des interactions entre les rôles peut aider à décrire les relations métiers entre les entités impliquées dans le processus de gestion de la souscription.

- Modèle de souscription : En nous basant sur les informations fournies par l'interface de souscription et sur les résultats fournis par le composant d'analyse, nous pouvons définir un ou plusieurs modèles de souscription.

4 LES INTERATIONS ENTRE LE SERVICE BROKER ET LES ROLES DE L'ECOSYSTEME

Cette section est consacré à la réalisation d'un scénario d'exécution qui montre en particulier les interactions entre les rôles de l'écosystème qui sont les clients et les fournisseurs d'application SaaS et notre service broker. La figure suivante présente les principales interactions entre les parties. Ce modèle est très semblable au modèle introduit par (Tian M. , Gramm, Ritter, & Schiller, 2004).

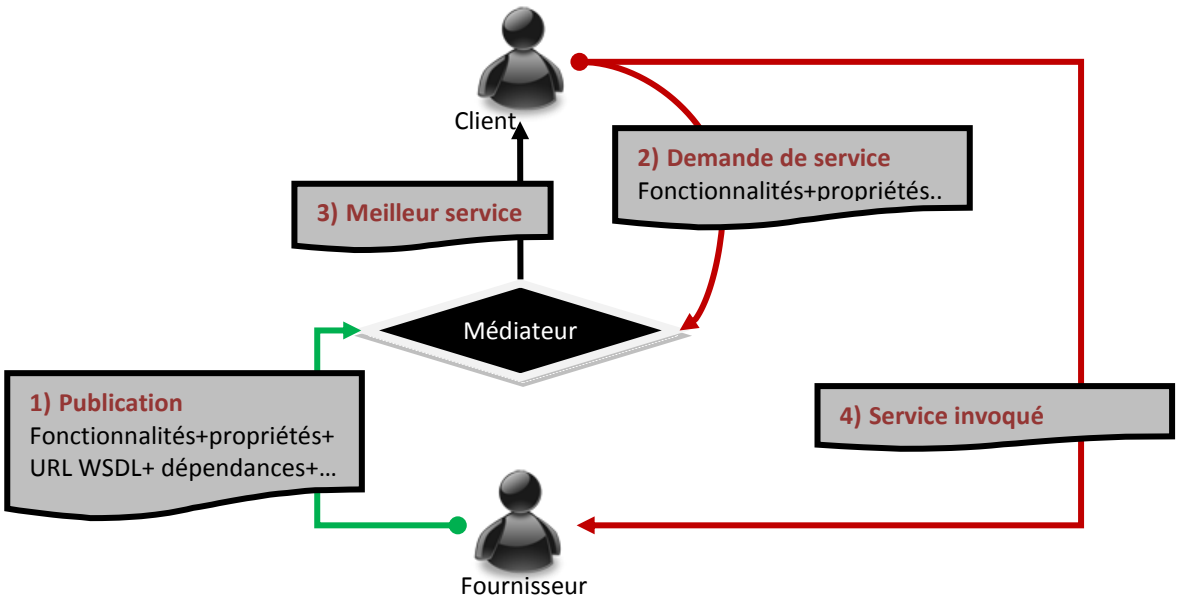


Figure 34. Interactions entre les composants.

Un fournisseur doit publier une application SaaS dans le broker en précisant l'URL de son WSDL et une liste des attributs d'interface. Il doit aussi définir les dépendances entre les modules fonctionnels de son offre.

Lorsqu'un client souhaite trouver une application X, il envoie une requête au broker en spécifiant la liste des modules fonctionnels et les préférences exigés. Après avoir faire une recherche, le broker lui retourne la liste des services trouvés. La liste retournée est ordonnée selon ses préférences. Le client invoquera un des services renvoyés pour s'abonner avec lui.

4.1 Publication

Lorsqu'un fournisseur souhaite publier une offre SaaS par le bien de notre SB, il doit interagir avec l'UDDI local au moyen d'une interface dédiée aux fonctions de publication. Ainsi, il doit définir l'URL du WSDL de son service plus les nivaux de compétence de chaque attributs d'interface. Comme nous avons vu dans le chapitre précédent, un fournisseur doit Spécifier :

- La liste des modules fonctionnels et leurs niveaux de compétence (voir chapitre 5 section 2.1).
- La liste des dépendances entre ces derniers pour choisir le patron de structure de service.
- Le niveau de compétence des autres paramètres tels que le coût, l'utilisabitilé, l'architecture, la configuration et personnalisation (chapitre 5 voir section2.2) Pour bien définir l'offre afin d'être mieux utiliser.

A. La fonction ***save_business*** permet de demander au broker d'enregistrer une entité métier. Nous détaillons la structure du message ***Save_business*** comme suit :

```
< save _business generic="2.0" xmlns="urn:uddi-org:api_v2">
    1) <authInfo>String</authInfo>
    2) <businessEntity>
            [<name/>…]
            [<address/>]
            [<contact/>]
       </businessEntity>
    3) < businesService>
            [<name/>…]
            [<Description />]
       < businesService/>
    4) <bindingTemplates>
       ……
            <URL ></URL >
            < Functionality >
                < Collaboration> PA_A1/P, PA_A2/P, PA_A3/P </Collaboration>
                < CM> PA_A1/P, PA_A2/P, PA_A3/P , PA_A4/P < /CM>
                < Suite bureautique> PA_A1/P, PA_A2/P, PA_A3/P </Suite bureautique>
                < CRM> PA_A1/P, PA_A2/P, PA_A3/P, PA_A4/P < /CRM>
                < BI> PA_A1/P, PA_A2/P, PA_A3/P, PA_A4/P , PA_A5/P < BI>
                < ERP> PA_A1/P, PA_A2/P, PA_A3/P, PA_A4/P , PA_A5/P, PA_A6/P </ ERP>
                < SCM> PA_A1/P, PA_A2/P, PA_A3/P, PA_A4/P , PA_A5/P </ SCM>
            < Functionality />
            <Propriety>
                <Reputation > PA_A1/P, PA_A2/P </Reputation >
                <Cout> PA_A1/P, PA_A2/P </Cout>
                <Utilisability> PA_A1/P, PA_A2/P, PA_A3/P, PA_A4/P , PA_A5/P, PA_A6/P </Utilisability>
                <Architecture> PA_A1/P, PA_A2/P, PA_A3/P </Architecture>
                <Configurability> PA_A1/P, PA_A2/P, PA_A3/P, PA_A4/P , PA_A5/P </Configurability>
            </propriety >
            <Dependances>… </Dependances>

    5) <categoryBag/>
    6) < publishAssertion />
</ save _business>
```

1. authInfo : contient le jeton d'authentification.
2. businessEntity : contient une description de l'entreprise offrant le service.
3. businesService : contient le nom, une description du service offert et l'url de son wsdl.
4. bindingTemplete : contiennent des informations techniques du service offert. Nous avons définit les balises *fonctionality* et *propriety* pour pouvoir spécifier le poids de tous les attributs définis au chapitre précédent. Nous avons défini aussi la balise *Dependances* permettant de définir les dépendances entre les modules fonctionnels du service offert afin de choisir un patron de structure de service.
5. L'élément categoryBag permet de spécifier une liste de localisateurs qui déterminent des références catégorielles propres aux entités ou services métier.
6. publishAssertion : met en correspondance deux ou plusieurs structures businessEntity.

B. Exemple d'un message ***Save_business***: notre exemple illustre un message ***Save_business*** de l'offre B donné en chapitre 5 (voir section 2.4).

```
< save _business generic="2.0" xmlns="urn:uddi-org:api_v2">
        <authInfo> ba748c9d-73ce-4cd9-85f8-294edddcbbf0 </authInfo>
        <businessEntity>
                <name> fourniseur2 </name>
                <address> </address>
                <contact></contact>
        </businessEntity>
        <businessEntity>
                <name> fournisseur1</name>
                <address></address>
                <contact></contact>
        </businessEntity>
        < businesService>
                <name>offer B</name>
                <Description >….</Description >

        < /businesService>
        <bindingTemplates>
        ……
                <URL > http://www.restfulwebservices.net/wcf/FedACHService.svc?wsdl </URL >
                < Functionality >
                        < Collaboration> 0.13, 0, 0.13 </Collaboration>
                        < CM> 0, 0, 0, 0 < /CM>
                        < Suite bureautique> 0, 0,0 </Suite bureautique>
                        < CRM> 0.25, 0.08, 0.08, 0.33 < /CRM>
                        < BI> 0, 0, 0, 0, 0 < BI>
                        < ERP> 0, 0, 0, 0, 0, 0 </ ERP>
                        < SCM> 0, 0, 0, 0, 0 </ SCM>
                < Functionality />
                <Propriety>
                        <Reputation > 0.12, 0.06 </Reputation >
                        <Cout> 0.03, 0.06 </Cout>
                        <Utilisability> 0.07, 0.02, 0.07, 0, 0.02, 0 </Utilisability>
                        <Architecture> 0.06, 0, 0.12 </Architecture>
                        <Configurability> 0.04, 0.04, 0.10, 0.04 , 0.15 </Configurability>
                </propriety >
                <Dependances>
                V |D M, CA |D M, CA |D V, CA |D AD , AD |D V ,Com |D M, M |D Com , Com |D V ,
                V |D Com , Com |D CA, CA |D Com , Com |D AD, AD |D Com, C|D M, M |D C , C |D
                V , V |D C , C |D CA, CA |D C , C |D AD, AD |D C
                </Dependances>
        </bindingTemplates>
        <categoryBag>…</categoryBag>
        < publishAssertion >…< /publishAssertion >
</ save _business>
```

4.2 Demande de service(Découverte)

Les clients interrogent le Broker pour demander une application effectuant une certaine fonctionnalité. Le client envoie deux critères au Service Broker.

- Le premier critère est une description de la fonction qu'il souhaite invoquer.
- Le second critère est optionnel. Il s'agit de la liste des paramètres supplémentaires du choix du client (voir chapitre 5 section 2.2).

La fonction find_business permet de rechercher un ou plusieurs éléments de type businessEntity en fonction de différents critères. Cette fonction renvoie un message businessList. En cas de recherche infructueuse par rapport aux critères de recherche utilisés, la liste renvoyée est vide. Dans le cas contraire, le message businessList retourné est constitué de structures de données de type businessInfo (forme abrégée d'une structure de données businessEntity), incluses dans une collection businessInfos.

A. Syntaxe du message find_business est la suivante :

```
<find_business [maxRows="nn"] generic="2.0" xmlns="urn:uddi-org:api_v2">
        [<name/> [<name/>]…]
        [<discoveryURLs/>]
        [<identifierBag/>]
        [<categoryBag/>]
        [<tModelBag/>]
        [<findQualifiers/>]
        <Parameters>
        < Functionality >
                < Collaboration> PA_A1/P, PA_A2/P, PA_A3/P </Collaboration>
                < CM> PA_A1/P, PA_A2/P, PA_A3/P , PA_A4/P < /CM>
                < Suite bureautique> PA_A1/P, PA_A2/P, PA_A3/P </Suite bureautique>
                < CRM> PA_A1/P, PA_A2/P, PA_A3/P, PA_A4/P < /CRM>
                < BI> PA_A1/P, PA_A2/P, PA_A3/P, PA_A4/P , PA_A5/P < BI>
                < ERP> PA_A1/P, PA_A2/P, PA_A3/P, PA_A4/P , PA_A5/P, PA_A6/P </ ERP>
                < SCM> PA_A1/P, PA_A2/P, PA_A3/P, PA_A4/P , PA_A5/P </ SCM>
        < Functionality />
        <Propriety>
                <Reputation > PA_A1/P, PA_A2/P </Reputation >
                <Cout> PA_A1/P, PA_A2/P </Cout>
                <Utilisability> PA_A1/P, PA_A2/P, PA_A3/P, PA_A4/P , PA_A5/P, PA_A6/P </Utilisability>
                <Architecture> PA_A1/P, PA_A2/P, PA_A3/P </Architecture>
                <Configurability> PA_A1/P, PA_A2/P, PA_A3/P, PA_A4/P , PA_A5/P </Configurability>
        </propriety >
        </Parameters>
</find_business>
```

1. L'attribut maxRows est optionnel et permet de limiter le nombre d'éléments renvoyés dans la liste résultante d'une requête de type find.
2. L'élément *name* précise un nom complet ou partiel d'entités métier à rechercher.
3. L'élément discoveryURLs spécifie une liste d'adresses URL associées aux entités métier recherchées. L'élément identifierBag permet de préciser une liste d'identificateurs métier qui correspondent aux entités métier ou services types recherchés.
4. L'élément *categoryBag* permet de spécifier la liste des références catégorielles propres aux entités ou services métier recherchés : la liste renvoyée comporte les structures de données qui sont classées dans chacune des catégories précisées.

5. L'élément *tModelBag* permet de spécifier une liste de clés de tModels. Ainsi, la liste renvoyée par la requête est filtrée et ne comporte que les entités métier ou les liaisons qui référencent l'ensemble des clés de tModels passées en paramètre (« et » logique). L'ordre des clés de tModels est indifférent.
6. L'élément *findQualifiers* est une collection de qualificateurs de recherche (ou filtres) qui peuvent être utilisés pour modifier le comportement standard d'une requête de type find. Cet élément est optionnel et peut être mis en œuvre conjointement à d'autres arguments de la requête.
7. L'élément *Parameters* précisé un des attributs définis en quatrième chapitre. La liste renvoyée par la requête contient les services métier classés en fonction des poids des différents attributs en appliquant l'algorithme détaillé en chapitre 5 (voir section 2.3).

B. Exemple d'un message *find_business*: notre exemple illustre un message find_business basé sur l'exemple donné en chapitre 5 (voir section 2.4).

```
<find_business [maxRows="nn"] generic="2.0" xmlns="urn:uddi-org:api_v2">
        <name> <name/>
        <discoveryURLs></discoveryURLs>
        <identifierBag></identifierBag>
        <categoryBag></categoryBag>
        <tModelBag></tModelBag>
        <findQualifiers></findQualifiers>

        <Parameters>
        < Functionality >
                < Collaboration> 0.13, 0, 0.20 </Collaboration>
                < CM>0, 0, 0, 0< /CM>
                < Suite bureautique> 0, 0, 0</Suite bureautique>
                < CRM> 0.19, 0, 0.09, 0.38 < /CRM>
                < BI> 0, 0, 0, 0, 0 < BI>
                < ERP>0, 0, 0, 0, 0, 0, 0 </ ERP>
                < SCM> 0, 0, 0, 0, 0 </ SCM>
        < Functionality />
        <Propriety>
                <Reputation > 0.10, 0.06 </Reputation >
                <Cout> 0.02, 0.06 </Cout>
                <Utilisability> 0.08, 0.04, 0.12, 0, 0, 0 </Utilisability>
                <Architecture> 0.05, 0, 0.11 </Architecture>
                <Configurability>0.03, 0.06, 0.10, 0, 0.13 </Configurability>
        </propriety >
        </Parameters>
</find_business>
```

C. **La Recherche des entités métier** : l'exemple ci-dessous est un programme java présenté dans le but de la recherche des entités métier en précisant une listes de critères ou de propriétés recherché :

```
import org.uddi4j.client.UDDIProxy;
import org.uddi4j.datatype.Name;
import org.uddi4j.response.*;
import org.uddi4j.transport.TransportFactory;
import java.util.Vector;

public class UDDIFindBusiness {
public static void main(String[] args) throws Exception {

…

Vector proprieties = new Vector();

proprieties.add( find_businnes.Propriety.GetPoid_attributs());

…

// la méthode select_business applique l'algorithme définit en chapitre 5 section 2.3 et retourne une liste ordonnée

BusinessList bl = proxy.select_business(null, null, null, null, null, null, proprieties, 0);

// Le code suivant est destiné à lister le résultat (noms et clés des entités métier //renvoyées) de la requête adressée à l'annuaire UDDI_loc.

BusinessInfos bis = bl.getBusinessInfos();
        if (bis.size() == 0) {
        System.out.println("no business(es) found");
        System.exit(0);
        }
System.out.println(bis.size()+" business(es) found\n");
Vector biv = bis.getBusinessInfoVector();
        for (int i = 0; i < biv.size(); i++) {
        BusinessInfo bi = (BusinessInfo)biv.elementAt(i);
        System.out.println(bi.getNameString());
        System.out.println(bi.getBusinessKey());
        System.out.println("\n");
        }
}
}
```

Cet exemple montre comment récupérer l'ensemble des entités métier en passant une liste des paramètres de choix.

4.3 Liste des applications SaaS retournées

Si le Broker n'a pas de services enregistrés dans l'UDDI interne qui peuvent répondre à la demande du client, il envoie une réponse avec un message vide. Sinon, il renvoie la liste des services existants répondants à la demande du client. Les services trouvés sont ordonnés selon les paramètres du choix de telle façon que le service le plus approprié soit le premier de la liste et ainsi de suite.

A. Le schéma de la structure de données businessInfo, contenue dans le message businessList, se présente ainsi :

```
<element name="businessInfo" type="uddi:businessInfo" />
<complexType name="businessInfo">
        <sequence>
                <element ref="uddi:name" maxOccurs="unbounded" />
                <element ref="uddi:description" minOccurs="0" maxOccurs="unbounded" />
                <element ref="uddi:serviceInfos" />
        </sequence>
        <attribute name="businessKey" type="uddi:businessKey" use="required" />
</complexType>
```

B. Apres avoir appliqué la méthode ***UDDIFindBusiness*** définit en section précédente (voir 3.2), et en prenant l'exemple détaillé en chapitre 5 (voir section 2.4), nous pouvons avoir le résultat suivant :

Offer B
ba748c9d-73ce-4cd9-85f8-294edddcbbf0

Offer C
d2033110-3aaf-11d5-80dc-002035229c64

Offre A
fdfdbba0-a7d3-11d5-a30a-002035229c64

4.4Invocation du service

En utilisant URL, le client peut invoquer le service choisi à partir de la liste des services trouvés. Ainsi, un formulaire de souscription doit être renvoyé à ce dernier. Comme c'est indiquer au chapitre précédent, notre broker aide les fournisseurs à choisir un modèle de souscription en se basant sur la structure de service et les interactions de l'écosystème.

La fonction *get_subscription_model* utilise les balises *Configurability* et *Dependances* de la fonction *save_business* d'une part, et la balise *Configurability* de la fonction *find_business* d'autre part pour sélectionner les patrons de souscription appropriés.

```
import org.uddi4j.client.UDDIProxy;
import org.uddi4j.datatype.Name;
import org.uddi4j.response.*;
import org.uddi4j.transport.TransportFactory;
import java.util.Vector;

public class get_subscription_model {
public static void main(String[] args) throws Exception {

String   model1= " " ;
String   model2= " " ;
String   model3=" ";
String   model4= " " ;
          if (Dependances != null ) {
          System.out.println("sélect Service Multiple Dépendant ");
          model1= "Service Multiple Dépendant" ;
          }
          else {
          System.out.println("sélect Service indépendants ");
          model2= "Service indépendants" ;
          }
          if ((fin_business.Parameters. Configurability[4]  = 0) or
          (save_business.Parameters. Configurability [4] = 0) ) {
          System.out.println("sélect self service");
          model3= "self service" ;
          }
          if ((fin_business.Parameters. Configurability[4] != 0) and
          (save_business.Parameters. Configurability[4]  != 0) )  {
          System.out.println("sélect hub-spoke ");
          model4= "hub-spoke" ;
          }
          if (save_business. businessEntity. length() ≥ 2 ) {
          System.out.println("sélect délégué");
          model4= "délégué" ;
          }
 Composition.compose( model1, model2, model3, model4);
}
}
```

5 CONCLUSION

Au cours de ce chapitre, nous avons démontré la nécessité d'avoir un environnement de confiance persistant pour les applications SaaS. Cela a été accompli par la conception et la mise en œuvre d'un service broker pour aider à gérer les communications et l'échange de données entre des clients qui souhaitent utiliser le système et les fournisseurs des applications SaaS qui souhaite exposer leurs services dans un environnement de confiance. Nous avons démontré également que le *service broker* permet aux utilisateurs d'accéder aux applications SaaS les mieux adaptées à leurs besoins. Il s'occupe des issues d'intégration et d'échange fiable de données provenant de différentes sources

autonomes. Il permet de mieux définir les applications afin d'être mieux exploitées. Et il donne la possibilité de générer un modèle de souscription pour ces dernières.

Nous avons présenté l'architecture détaillée de notre Service Broker ainsi que toutes les composantes des différents unités qui sont : unité d'interface, unité d'UDDI, unité de WSDL, unité d'interopérabilité et enfin l'unité de souscription. Nous avons présenté aussi les différentes interactions entre les composantes de notre intergiciel. Nous avons présenté par la suite un exemple d'interactions entre les composants de l'écosystème et notre broker.

CONCLUSION ET PERSPECTIVES

Nous avons vu qu'au-delà de l'effet de mode autour de la notion de Cloud Computing, le SaaS (Software as a Service) s'inscrit dans l'évolution de fond de l'informatique et des systèmes d'information d'entreprise. Le modèle SaaS se développe rapidement et gagne des parties stratégiques du système d'information dans des entreprises petites et moyennes. Il s'agit d'une tendance qui va certainement modifier la manière d'intégrer des solutions technologiques dans l'entreprise. C'est donc dans cette optique que notre travail s'inscrit.

Pour cela, nous avons cherché à comprendre la notion du Cloud Computing à travers l'évolution historique des systèmes informatiques, à définir ses différentes formes et de spécifier ses caractéristiques. Nous avons étudié par la suite les applications SaaS. Nous nous sommes concentrés sur certains attributs qui définissent et distinguent une application SaaS des autres applications. Nous nous sommes intéressés particulièrement à l'architecture impliquée dans le développement des applications SaaS. Nous avons abordé quelques issus des applications développées sous ce nouveau paradigme telles que la personnalisation, le multi-client, l'extensibilité, la scalabilité des données…etc. Nous avons établi également une comparaison entre les SaaS et les types d'application hébergées ainsi que les différents avantages et inconvénients de ce nouveau concept.

Nous avons vu que nombreux sont les articles qui ont traité les applications SaaS, mais très peu se sont focalisés sur la construction d'une couche intermédiaire pour faciliter la découverte des applications SaaS les plus adaptées aux besoins du client et la souscription à ces dernières. Une couche qui permet aux fournisseurs aussi de publier leurs applications SaaS. Elle leur donnerait la possibilité de bien décrire leurs produits afin d'être bien utilisés.

C'est la raison pour laquelle nous avons développer deux méthodes à savoir la méthode de sélection d'un produit SaaS le mieux adapté aux besoins des clients et la méthode de génération de modèle de souscription. Nous avons proposé plusieurs facteurs pour le choix de SaaS tels que : les fonctionnalités, l'architecture, l'utilisabilité, la réputation du fournisseur, le coût et la configurabilité et la personnalisation. Nous avons esquissé un algorithme de sélection d'un SaaS. Nous avons modifié le modèle de souscription basé sur les patrons qui a été proposé par (Jiang, Sun, Tang, Snowdo, & Zhang, 2009)et nous avons proposé des expressions et un formalisme plus général.

En dernier lieu, nous avons mis en œuvre d'une architecture détaillé de l'intergiciel qui résout quelques problèmes actuels des applications SaaS. Cet intergiciel (service broker) permet à la fois la publication et la sélection d'un modèle de souscription relatives aux applications SaaS pour le fournisseur et la recherche et la sélection du produit SaaS le mieux adapté pour le client. La technologie de base derrière cet intergiciel est les services web.

Comme perspectives, nous identifions les directions suivantes :

- Définir un modèle pour l'identification d'objectifs d'amélioration des applications en se basant sur l'analyse comparative et en utilisant les niveaux de compétence
- Proposer un analyseur d'applications qui peut évaluer d'une façon automatique les applications SaaS en affectant les valeurs des poids aux différents attributs.
- Alléger les interactions et la saisie de l'utilisateur en développant des outils de sauvegarde des profils et des préférences d'une part et d'analyse d'applications et de WSDL pour tiré le maximum d'informations d'une autre part.

- Améliorer le composant d'interopérabilité pour une meilleure communication est intégration des données.

- Etudier la composition des SaaS et développer un composant qui permet la construction de nouveaux logiciels en fonction des exigences des clients.

BIBLIOGRAPHIE

- Abramson, D., Buyya, R., & Giddy, J. (2002, oct). A Computational Economy for Grid Computing and its Implementation in the Nimrod-G Resource Broker. *Future Generation Computer Systems 18* , pp. 1061–1074.
- Anderson, C. (2006). The long tail [M]. *BeiJing: ChINA CITIC PRESS* .
- Bai, X. (2010, avril 26). *Collaborative testing of SaaS.* Retrieved janvier 2011, from http://itechs.iscas.ac.cn/cn/news/first_iscas_swell_joint_workshop/Collaborative%20Testing%20of%20SaaS-XiaoyingBai.pdf
- Benlian, A., Hess, T., & Buxmann, P. (2009, mais). Drivers of SaaS-Adoption – An Empirical Study of Different Application Types. *BISE "Business &Information Systems Engineering "* , pp. 357-369.
- Bennett, K. H., & al. (2003). A Broker Architecture for Integrating Data Using a Web Service Environment. *In Proc. of the International Conference on Service-Oriented Computing (ICSOC2003), LNCS 2910* , pp. 409-422.
- Boucher, E. (2009, Avril). Software as a Service. *Thèse* . Paris, France: Mines Pari- ParisThec.
- Buyya, R., & Venugopal, S. (2005, july). A Gentle Introduction to Grid Computing and Technologies. *computer society of india* .
- Buyya, R., Yeo, C. S., Venugo, S., Broberg, J., & Brandic, I. (2008, Sept). Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. *In Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC 2008), Dalian* .
- Buyya, R., Yeo, C. S., Venugo, S., Broberg, J., & Brandic, I. (2008). Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility.
- Cao, L., & Zhou, G. (2009). Analysis of SaaS-Based Informationization in Small and Medium-Sized Logistics Enterprises. *Third International Symposium on Intelligent Information Technology Application Workshops-IEEE* , pp. 78-81.
- Chang, j. G., Wei, S., Ying, H., Zhi, H., & Bo, G. (2007, août 8). A Framework for Native Multi-Tenancy Application Development and Management. *The 9th IEEE International Conference on E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE 2007).*
- Chauvet, J. (2002). *Services Web avec SOAP, WSDL, UDDI, ebXML...* EYROLLES.
- Chelbabi, M. (2006). *Découverte de Services Web Sémantiques une Approche basée sur le Contexte.*
- Chong, F., & Carraro, G. (2006, April). *Architecture Strategies for Catching the Long Tail.* Retrieved from MSDN architecture center: http://msdn2.microsoft.com/en-us/library/aa479069(printer).aspx
- Chong, F., Carraro, G., & Wolt, R. (2006, June). *Multi-Tenant Data Architecture*. Retrieved from msdn: http://msdn.microsoft.com/en-us/library/aa479086.aspx
- Choudhary, V. (2007). Software as a Service: Implications for Investment in Software Development. *Proceedings of the 40th Hawaii International Conference on System Sciences -IEEE* .
- Chui, M., & al, e. (2009, Fevrier). Six ways to make Web 2.0 tools work. *McKinsey quarterly* .
- Degwekar, S., Su, S. Y., & Lam, H. (2004., July 6-9). Constraint Specification and Srocessing in Web Services Publication and Discover. *Proceedings of the IEEE International Conference on Web Services* , pp. 210 – 217.
- Desisto, R. (2009, January 16). the Five Most-Common SaaS Assumptions,. *Gartner Fact* .
- Dhesiaseelan, A., & Ragunathan, V. (2004, July). Web Services Container Reference Architecture (WSCRA). *IEEE International Conference on Web Services (ICWS'04) , 6-9* (00), pp. 806 – 807.
- Dubey, A., & Wagle, D. (2007, June). Delivering software as a service. *The McKinsey Quarterly* .
- Eisentraut, R., Koch, M., & Mösle, K. (2001). Building trust and reputation in communities and virtual enterprises. *Proceedings of the Seventh Americas Conference on Information Systems* , pp. 1506-1509.
- Ferris, C., & Farrell, J. (2003). What Are Web Services? *Comm. ACM , 46* (6), p. 31.

- Guo, C. J., Sun, W., Huang, Y., Wang, Z. H., & Gao, B. (2007). A Framework for Native Multi-tenancy Application Development and Management. *CEC* .
- Howard, R., & Kerschberg, L. (2004, Aug 3). A knowledge-based Framework for Dynamic Semantic Web Services Brokering and Management. *Proceedings of the 15th annual Database and Expert Systems Applications* , pp. 174 – 178.
- Huhns, M. N., & Singh, P. M. (2005, Jan/Feb). Service-Oriented Computing: Key Concepts and Principles. *IEEE Internet Computing , 09* (1), pp. 75-81.
- Jiang, Z., Sun, W., Tang, K., Snowdo, J., & Zhang, X. (2009, July 6-10). A Pattern-based Design Approach for Subscription Management of Software as a Service. *Proceedings of the 2009 Congress on Services – I* , pp. 678-685.
- Jiehui, J., Ya, W., Jianqing, F., Jiyi, W., & Zhijie, L. (2010). Research on Key Technology in SaaS. *2010 International Conference on Intelligent Computing and Cognitive Informatics* , pp. 384-387.
- Knorr, E. (2006). *Software as a Service: The Next Big Thing.* Retrieved from http://www.infoworld.com/article/06/03/20/76103_12FEsaas_1.html
- Laplante, P. A., Zhang, J., & Voas, J. (2008, May/June). Distinguishing between SaaS and SOA. *Services Computing, IEEE Computer Society* , pp. 46-50.
- Lee, J. C.-K., Mohapi, S., & Hanrahan, H. E. (2001). Service Subscription Information Management in a TINA Environment using Object-Oriented Middleware. *Intelligent Network Workshop 2001* .
- Lee, J. Y., Lee, J. W., Cheun, D. W., & Kim, S. D. (2009). A Quality Model for Evaluating Software-as-a-Service in Cloud Computing. *Seventh ACIS International Conference on Software Engineering Research, Management and Application-IEEE* , pp. 261-266.
- Liao, H., & Tao, C. (2008). An Anatomy to SaaS Business Mode Based on Internet. *International Conference on Management of e-Commerce and e-Government* (pp. 215-220). IEEE Computer Society.
- Limam, N., & Boutaba, R. (2010, Jan 6). Assessing Software Service Quality and Trustworthiness at Selection Time. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING , XX* (36), pp. 1-16.
- Liu, G. (2010). Research on Independent SaaS Platform. *IEEE* .
- Liu, X. Z., Huang, G., & Mei, H. (2006). Towards Service Discovery and Subscription based on Community-of-Interest. *SOSE 2006* .
- Liu, X. Z., Zhou, L., Huang, G., & Mei, H. (2007). Consumer-Centric Web Services Discovery and Subscription. *ICEBE 2007* .
- Lotus Development Corporation. (1995). *Groupware: Communication, Collaboration and Coordination.* Retrieved 2011, from Intranet Journal: http://www.intranetjournal.com/faq/lotusbible.html
- Lu, Y., & Sun, B. (2009). The Fitness Evaluation Model of SAAS for Enterprise Information System. *IEEE International Conference on e-Business Engineering* , pp. 507-511.
- luitinfotech. (n.d.). *saas-asp-difference.pdf.* Retrieved 04 15, 2011, from luitinfotech: www.luitinfotech.com/kc/saas-asp-difference.pdf
- Ma, D., & Seidmann, A. (2008). The Pricing Strategy Analysis for the Software-as-a-Service Business Model. *The 5th International Workshop on Grid Economics and Business Models* .
- Macquet, L. (2009, Mars 17). *panorama du SaaS/PaaS/IaaS.* Retrieved 2010, from http://www.01net.com/Pdf_DSI/3_Pres_Laurent_MACQUET_Logica_Panorama_du_Saas.pdf
- Maesano, L., Bernard, C., & Le Galles, X. (2003). *Services Web avec J2EE et .NET. Conception et implémentations.* EYROLLES.
- Manish, G., & Shrikant, M. (2009). An Approach for Selecting Software-as-a-Service (SaaS) Product. *IEEE International Conference on Cloud Computing* , pp. 155-185.

- Mietzner, R., Leymann, F., & Papazoglou, M. P. (2008). Defining Composite Configurable SaaS Application Packages Using SCA,Variability Descriptors and Multi-Tenancy Patterns. *The Third International Conference on Internet and Web Applications and Services* (pp. 156-161). IEEE Computer Society.
- Moore, B., & Qusay, H. M. (2009, May 10-13). A Service Broker and Business Model for SaaS Applications. *AICCSA 2009. IEEEACS International Conference on Computer Systems and Applications* , pp. 322-329.
- Newcomer, E. (2002). *Understanding Web Services- XML, WSDL, SOAP and UDDI.* Addison Wesley Professional.
- Nitu. (2009, February 23–26). Configurability in SaaS (Software as a Service) Applications. *ACM, ISEC'09* , pp. 19-26.
- Pervez, Z., Lee, S., & Lee, Y.-K. (2010, fevrier 7-10). Multi-Tenant, Secure, Load Disseminated SaaS Architecture. pp. 214-219.
- Pfister, G. (2003). Cluster Computing. *Encyclopedia of Computer Science, The ACM Digital Library* , pp. 218-221.
- Plouin, G. (2009). *Cloud Computing et SaaS.* Donod.
- Pressman, R. (1996-2001). Software Engineering: A Practitioner's Approach. *McGraw-Hill Science* .
- Ralston, A., Reilly, E. D., & Hemmendinger, D. (2003). Encyclopedia of Computer Science, 4th edition. *John Wiley and Sons Ltd. Chichester, UK* .
- Rohleder, C., Davis, S., & Günther, H. (2005). SOFTWARE CUSTOMIZATIONWITH XML. *Issues in Information Systems , VI* (2), pp. 345-351.
- Saaty, T. L. (1990, September). How to make a decision: The Analytic Hierarchy Process. *European Journal of the Operational Research , 48* (1, 5), pp. 9-26.
- Saaty, T. L., & Vargas, L. G. (2001). Models, methods, concepts & applications of the Analytical Hierarchy Process. *Kluwer Academic Publisher* .
- Salesforce.com. (n.d.). *Introduction to the Apex Platform.* Retrieved 2011, from ThecWorld: http://media.techworld.com/cmsdata/whitepapers/4696/creating_apps_appexchange.pdf
- Schadler, T. (2009, janvier 27). Should your email live in the cloud? A comparative cost analysis. *Forrester Research* .
- Sheryl, K. (2004, June). Understanding Total Cost of Ownership of a Hosted vs. Premises-Based CRM Solution. *Yankee Group Repor* .
- Smith, K. T., Daconta, M. C., & Obrst, L. J. (2003). *The Semantic Web : A Guide to the Future of XML, Web Services, and Knowledge Management.* Wiley, John and Sons, Incorporated.
- Stamford, C. (2009, July 9). *Gartner Says Cloud Consumers Need Brokerages to Unlock the Potential of Cloud Services.* Retrieved 2010, from Gartner: http://www.gartner.com/it/page.jsp?id=1064712
- Sun, W., Zhang, X., Guo, C. J., Sun, P., & Su, H. (2009). Software as a Service: Configuration and Customization Perspectives. *Services 2009* .
- Sun, W., Zhang, X., Guo, C. J., Sun, P., & Su, H. (2009). Software as a Service: Configuration and Customization Perspectives. *Services 2009* .
- Sun, W., Zhang, X., Jie, C. G., Sun, P., & Su, H. (2008). Software as a Service: Configuration and Customization Perspectives. *IEEE Congress on Services Part II* , pp. 18-24.
- Symons, C. (2004). The Balanced Scorecard, an IT Perspective. *Forrester Research* .
- Tian, M., Gramm, A., Ritter, H., & Schiller, J. (2004, Sep). Efficient Selection and Monitoring of QoS-Aware Web Services with the WS-QoS Framework. *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence(WI'04)* , pp. 152 – 158.

- Tian, M., Gramm, T., Naumowicz, A., Ritter, H., & Schiller, J. (2003, Dec). A Concept for QoS Integration in Web Services. *Proceedings. Fourth International Conference on Web Information Systems Engineering Workshops* , pp. 149-155.
- Turner, M., Budgen, D., & Brereton, P. (2003, oct). Turning Software into a Service. *Computer, IEEE Computer Society , 36* (10), pp. 38-44.
- Turner, M., Zhu, F., Kotsiopoulos, I., Russell, M., Budgen, D., Bennett, K., et al. (2004, May). Using Web Service Technologies to Create an Information Broker: An Experience Report. *Proceedings 26th International Conference on Software Engineering 2004* , pp. 552 – 561.
- Weier, M. H., & Smith, L. (2007, Apr 16). Businesses Get Serious about Software as a Service. *Information Week* , pp. 46–48.
- Weiss, T. (2010, January 28). *Getting Started with ABAP.* Retrieved from SAP Community Network: http://www.sdn.sap.com/irj/sdn/abap?rid=/webcontent/uuid/90e7556d-ed76-2910-1592-b6af816225cc
- Wittgreffe, J. (2010). *Cloud Service Broker.* Retrieved from http://www.tmforum.org/CloudServiceBroker/8437/home.html
- Yates, S. (2009, fevrier 10). Cloud Computing Can Save You Money. *Forrester Research Inc* .
- Yu, B., Liu, Y., Hu, J., & Lin, K. (2009). A Service-Oriented CRM For SMEs Based On The Enterprise Level Customer. *2009 IEEE* .
- Zeng, L., Benatallah, B., Lei, H., Mgu, A., Flaxer, D., & Chang, H. (2003, June). Flexible Composition of Enterprise Web Services. *Electronic Markets , 13* (2), pp. 141-152.
- Zhang, L.-J., Zhang, J., & Cai, H. (2007). Services Computing. *Springer* .

Webographie

- http://code.google.com/intl/fr/appengine/docs/whatisgoogleappengine.html.
- *http://conscience-sociale.blogspot.com/2008/07/la-guerre-dans-les-nuages.html*
- *http://www.commentcamarche.net/contents/web-services/soa-architecture-orientee-services.php3*
- http://aws.amazon.com/ec2
- http://aws.amazon.com/s3
- http://www.atelier.fr/informatique/10/05012009/Cloud-Computing-grid-architecture-amazon-flexibilite-externaliser-37655-.html
- http://www.itrmanager.com/tribune/214/Cloud-Computing-saas-informatique-durable-br-jean-louis-melin-alaloop.html
- http://www.silicon.fr/fr/news/2008/08/25/ibm___lotus_foundation_et_le_saas_bluehouse_
- http://www.ei-technologies.com/detail.do?noArticle=120&idRubrique=48
- http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci213801,00.html
- http://blog.aysoon.com/Que-sont-les-applications-riches-RIA-Partie-1-Definition-et-usages-195
- http://ajaxmar08.sys-con.com/node/609607 ; http://www.asp.net/ajax/ ;
- http://ajaxoct07.sys-con.com/
- http://sites.google.com/site/leweb20/ ; http://fr.wikipedia.org/wiki/Web_2.0
- http://fr.wikipedia.org/wiki/Grappe_de_serveurs
- *http://www.tinac.com/specifications/documents/comp.pdf*
- *http://www.tinac.com/specifications/documents/overall.pdf*
- *http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm*
- *http://ComputingintheCloud.wordpress.com/2008/09/25/utility-Cloud-Computingflashback-to-1961-prof-john-mccarthy/*

- *http://www.ei-technologies.com/detail.do?noArticle=120&idRubrique=48;*
- *http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci213801,00.html*
- *http://blog.aysoon.com/Que-sont-les-applications-riches-RIA-Partie-1-Definition-et-usages-195*
- *http://ajaxmar08.sys-con.com/node/609607*
- *http://www.asp.net/ajax/*
- *http://ajaxoct07.sys-con.com/*
- *http://sites.google.com/site/leweb20/*
- *http://fr.wikipedia.org/wiki/Web_2.0*
- *http://fr.wikipedia.org/wiki/Grappe_de_serveurs*
- *http://fr.wikipedia.org/wiki/Representational_State_Transfer*
- *http://aws.amazon.com/ec2 ; http://aws.amazon.com/s3*
- *http://www.silicon.fr/fr/news/2008/08/25/ibm___lotus_foundation_et_le_saas_bluehouse_*