1.1Introduction

Les systèmes informatiques distribués recouvrent un champ très vaste dont font partie les grilles de calcul, le stockage d'informations dans des réseaux de pairs, le déploiement d'applications web. SOC, acronyme de Service Oriented Computing, désigne ce paradigme de programmation émergeante, pour le développement de systèmes d'informations distribués, dans lequel les concepts de distribution, d'ouverture, de messagerie asynchrone et de faible couplage tiennent un rôle primordial.

Les besoins des SOC pour la satisfaction des précédents cas d'usage peuvent être plus facilement satisfaits à travers une architecture répondant aux propriétés requises. Une telle architecture est appelée SOA ou Architecture Orientée Services.

Les Services Web comme une instanciation de SOA représentent une technologie récente utilisée pour supporter le développement des systèmes d'information distribués, en particulier les applications B2B sur Internet. Aujourd'hui, ils semblent être la solution la plus adaptée pour assurer l'interopérabilité sur Internet.

L'objectif de ce chapitre est d'introduire le concept de SOA et les services web.

1.2 L'Architecture Orientée Services (SOA)

1.2.1 Concepts de base

Les architectures orientées services sont le fruit d'une lente maturation technologique s'appuyant sur les travaux précurseurs à la base des notions d'objets et de composants. Une compréhension préalable de ces concepts est nécessaire à l'obtention d'une plus fine appréhension des SOAs. [01]

1.2.1.1 Objet :

La complexification des systèmes logiciels a entraîné dans son sillage l'ingénierie logicielle vers un objectif de rationalisation de la production. Ainsi, de par leur évolution constante et l'augmentation de leur distribution sur le réseau, ces systèmes nécessitent la mise en place de paradigmes permettant d'assurer un certain niveau de qualité et fiabilité, tout en offrant une plus grande flexibilité et réutilisabilité.

L'approche à objets [Jacobson, 1991] s'est indubitablement révélée comme un standard de facto parmi les différents paradigmes de développement logiciel. Cette approche a été rendue

possible par l'adoption intrinsèque du paradigme objet au cœur de langages tels Small talk, Java ou, dans une moindre mesure, C++. L'innovation portée par les objets a consisté à regrouper sous un même concept un ensemble de principes préexistants tels que :

- **L'encapsulation**, l'interface d'un objet est bien distincte de son implantation, et son comportement exprimé via les méthodes de cette interface ;
- **L'héritage**, c'est-à-dire de la capacité à définir les caractéristiques d'un objet par extension ;
- **Le polymorphisme**, il est possible d'interagir de la même façon avec des objets de formes différentes.

1.2.1.2 Composant :

Par rapport aux objets, le paradigme de composant représente une augmentation du niveau d'abstraction lors de la conception logicielle. En ce sens, le composant ne remplace pas l'objet, il se construit même bien souvent autour, dans les langages de programmation courants (par exemple Java). Il répond à un besoin concret car, si l'utilisation exclusive d'objets lors de la programmation de systèmes simples semble raisonnable, la programmation d'applications réparties de plus grande ampleur nécessite une augmentation du niveau d'abstraction de leurs briques de base, afin de maintenir leur complexité relativement appréhendable.

Un composant dispose d'interfaces spécifiées à l'aide de contrats, dans l'optique d'une composition ultérieure à sa conception, par des entités tierces. L'implantation de ce paradigme par de nombreuses technologies telles que les Enterprise Java Beans (EJB) de Sun, COM de Microsoft, ou FRACTAL du consortium OW2, a permis leur large diffusion.

Ces technologies déchargent le plus souvent le développeur de la gestion de certaines problématiques de mise en œuvre telles que le support de la sécurité, de la persistance et des transactions.

Ces mécanismes de support à l'exécution des composants seraient difficiles à mettre en place dans des environnements plus faiblement couplés (tels les services, détaillés à la sous-section suivante).

La contrepartie est que les composants sont souvent bien trop liés à leurs plateformes de support respectives pour être efficacement interopérables.

1.2.1.3 Service

Dans la lignée de l'approche à objets ou à composants, le paradigme de service tend à fournir un niveau d'abstraction encore plus élevé lors du développement de systèmes informatiques en général, et des applications réparties en particulier. Cette augmentation du niveau d'abstraction passe par une adoption poussée du concept d'encapsulation des fonctionnalités et de celui de la réutilisabilité des services existants. Les services répondent ainsi directement aux problématiques d'intégration logicielle dans les applications réparties modernes.

Tout comme un composant, un service est défini comme étant une unité logicielle autonome. Ceci est d'autant plus vrai pour les services Web qui sont définis comme étant sans état ("stateless"). Les services se distinguent cependant des composants dans la mesure où leur définition met en avant une indépendance poussée par rapport aux plateformes, et une interopérabilité élevée. Voyons maintenant comment faire interagir ces services au sein d'un Environnement. On parle alors d'architecture orientée service [01].

1.2.2 Pourquoi SOA ?

Au début les départements informatiques ont tenu un point de vue orienté application du monde. Ils ont alloué la plupart de leurs budgets à l'achat, le déploiement et la maintenance des applications. À l'époque, un grand pourcentage de budget a été alloué aux projets d'intégration pour fournir un soutien plus large et plus transparent pour les processus d'affaires. En effet, ces projets n'ont souvent pas atteint leurs promesses en termes de la rigidité de l'intégration câblés. Tout changement à une seule application rend le retour en arrière de l'infrastructure très coûteux.

Après que les techniques conventionnelles de programmation ont été remplacées par le développement de modèle à objet, la programmation par composant est apparue et a permis un développement des applications basé sur cette technique de programmation, un nouveau concept architecturel est apparu : l'architecture orienté service ou SOA (Service Oriented Architecture).

En quelques années, SOA est devenue un thème majeur pour les systèmes d'informations d'entreprise. Plus qu'une nouvelle technologie ou méthode, c'est la convergence de plusieurs approches existantes, et l'émergence d'une forte adhésion des

directions informatique et métier à un même objectif : une meilleure agilité des systèmes face aux transformations nécessaires. La figure 1.1 représente une architecture basique du SOA.

Cette architecture est apparue pour :

- Gérer l'intégration entre des applications dans le SI.
- Construire un SI plus flexible.
- Faciliter l'innovation.
- Réduire les coûts.
- Réagir rapidement à tous les changements.
- Réduire les doublons entre applications [02].

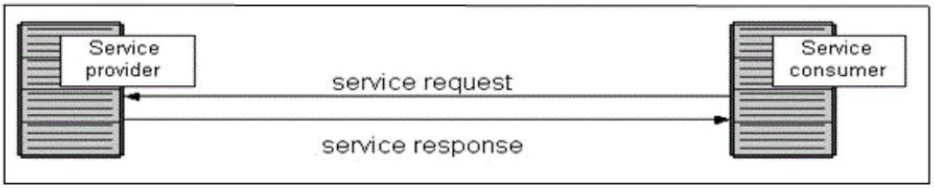

***Figure 1.1 : «Architecture basique de SOA»* [02]**

1.2.3 Définition d'architecture orientée service

On trouve plusieurs définitions de l'architecture SOA dans la littérature on a choisi les suivantes : Nicolai M.Josuttis **:** *« SOA is not a concrete architecture it is something that leads to a concrete architecture. You might call it a style, paradigm, concept, perspective, philosophy, or representation. That is, SOA is not a concrete tool or framework you can purchase. It is an approach, a way of thinking, a value system that leads to certain concrete decisions when designing concrete software architecture. »*

Nicolaï M.Josuttis veut dire que SOA n'est pas une architecture, mais c'est un paradigme qui nous conduit à une architecture. C'est une approche, une voie de pensée qui nous conduit à des certaines décisions.

OASIS : *« A SOA ecosystem is a network of discrete processes and machines that, together with a community of people, creates, uses, and governs specific services as well as external suppliers of resources required by those services. »*

Après la traduction on extrait qu'un écosystème SOA est un réseau de processus et de machines discrètes qui crée, utilise et réagit des services spécifiques ainsi que des fournisseurs externes de ressources requises par ces services.

Thomas Erl : *« SOA establishes an architectural model that aims to enhance the efficiency, agility, and Productivity of an enterprise by positioning services as the primary means through which solution logic is represented in support of the realization of strategic goals associated with service-oriented computing. »*

Après la traduction, SOA est un modèle d'architecture. Son but est de progresser l'efficacité, l'agilité et la production en positionnant les services comme le primaire, c.-à-d. avec quel solution est représenté dans un support de la réalisation des buts stratégiques associées avec SOA. [03].

Todd Biske : *« A SOA, therefore, is quite simply, an architecture that utilizes the core concepts of service providers and service consumers to define a system. »*

Comme Todd Biske a dit que cette architecture est vraiment simple. Elle utilise le cœur des concepts de deux services : le fournisseur et le consommateur pour définir un système.

SOA est un style d'architecture destiné à fournir un couplage lâché entre les composants d'un système. Un service est une fonction métier sans état qui accepte des demandes et renvoie des réponses à travers une interface bien définie.

Cependant, l'aspect le plus important de l'architecture SOA est qu'elle permet de séparer L'implémentation du service de son interface. C'est cette caractéristique qui assure le haut degré. [03].

1.2.4 Principaux objectifs de l'architecture orientée services

On dénombre trois grands objectifs de l'architecture orientée services, chacun axé sur une partie distincte du cycle de vie applicatif [04].

Le premier vise à structurer sous forme de services les procédures ou composants logiciels. Ces services sont conçus pour être faiblement couplés aux applications : ils ne servent qu'en cas de besoin. Ils sont prévus pour que les développeurs, tenus de standardiser la création de leurs applications, les utilisent facilement.

Le deuxième objectif est de fournir un mécanisme de publication des services disponibles qui comprend la fonctionnalité et les besoins d'entrée/sortie (E/S ou I/O). Les services sont publiés de manière à faciliter leur intégration aux applications.

Le troisième objectif de l'architecture SOA est de contrôler l'utilisation de ces services pour éviter tout problème de sécurité et de gouvernance. La sécurité de cette SOA est surtout axée

sur la sécurité des composants individuels en son sein, sur les procédures d'authentification et d'identification en lien avec ces composants, et la sécurisation des connexions entre les composants de l'architecture**.**

1.2.5 Les rôles dans une architecture orientée services

Les différents rôles dans une architecture orientée services sont les suivants [05]:

- **Le consommateur de services** est une application, un module logiciel ou un autre service exigeant un service. Il poste une requête à l'annuaire de services, et communique avec le service au moyen de la couche transport, puis exécute la fonction du service. Le consommateur de services exécute le service selon le contrat d'interface prédéfini.
- **Le fournisseur de services** est une entité accessible via le réseau acceptant et exécutant des demandes de consommateurs. Il édite ses services et contrats d'interface à l'annuaire de services de sorte que le consommateur de services puisse découvrir et accéder au service.
- **Un annuaire de services** est un élément essentiel pour la découverte de services. Il contient un répertoire de services disponibles et permet la consultation des interfaces de services par les consommateurs intéressés.

Chaque entité présente dans une architecture orientée services peut jouer un (ou plusieurs) des trois rôles décrits ci-dessus. Les opérations possibles dans une architecture orientée services sont [05] :

- **Edition/Publication** : pour être accessible, une description de service doit être publiée de sorte qu'elle puisse être découverte et appelée par un consommateur de services.
- **Découverte** : un demandeur de services localise un service via une requête à l'annuaire de services portant sur les critères du service recherché.
- **Liaison/Invocation** : après la recherche de la description de service, le consommateur de services procède par appel du service selon les informations relatives à la description du service.

Les objets clefs dans une architecture orientée services sont :

- **Les services** : un service rendu disponible via la publication de la description de son interface lui permet d'être appelé par un consommateur de services.
- **La description des services** : une description de service indique la manière dont le consommateur de services agira avec le fournisseur de services. Elle spécifie le format de la demande et de la réponse du service. Cette description peut indiquer un ensemble de conditions préalables, de post-conditions et/ou de qualité de service **(QoS)**.

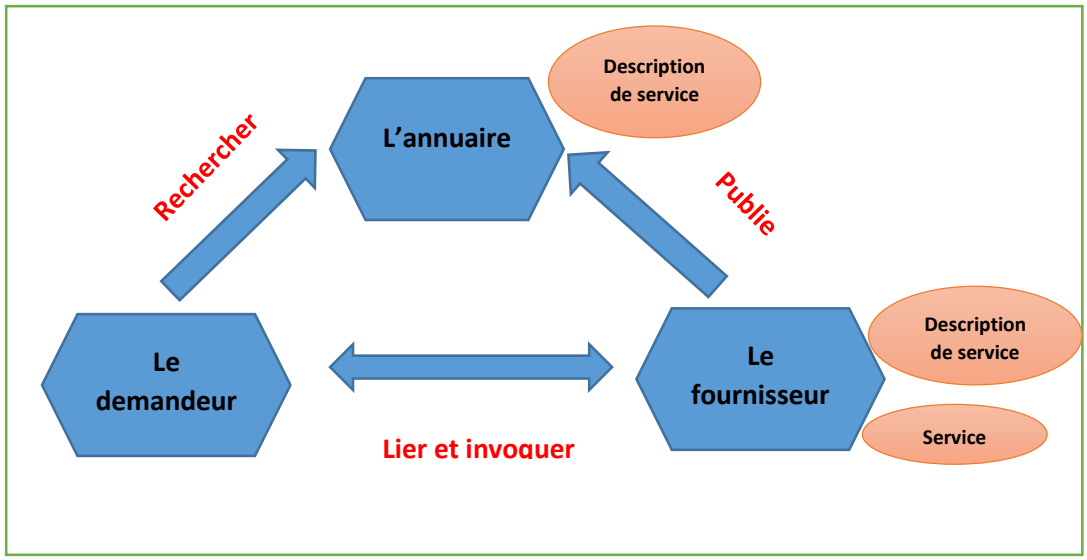

Figure 1.2. : «Interaction principales SOA»

1.2.6 Les principes de SOA

Huit principes sont définis par Thomas Erl [06] comme suit :

1.2.6.1 Contrats de services standardisés :

- Les services expriment leur but, leurs fonctionnalités, et leurs types de données via un contrat de service.
- Il adhère à un accord de communication standard, qui est défini collectivement par un ou plusieurs documents de description de service.

1.2.6.2 Couplage libre (niveau de dépendance) :

Une relation spécifique à l'intérieur et à l'extérieur du service, en mettant l'accent sur la réduction des dépendances entre :

- Le contrat de service, son implémentation et ses consommateurs.

1.2.6.3 Abstraction de services :

Les contrats de service ne contiennent que des informations essentielles et l'information sur les services est limitée à ce qui est publié dans les contrats de services.

1.2.6.4 Réutilisabilité des services :

Les services sont conçus pour être réutilisés. Ces services réutilisables sont conçus de manière à ce que leur solution logique soit indépendante d'un processus métier ou d'une technologie particulière.

1.2.6.5 Autonomie des services :

Les services exercent un niveau élevé de contrôle sur leur environnement d'exécution sous-jacent.

1.2.6.6 Capacité de composition en modules de services :

Les services sont des participants efficaces de la composition, quelle que soit la taille et la complexité de la composition.

1.2.6.7 Services sans états :

La gestion d'informations de l'état peut compromettre la disponibilité d'un service et compromettre son potentiel d'évolution. Les services sont donc idéalement conçus pour prendre un état sauf si cela est vraiment nécessaire.

Réduction des ressources consommées par un service.

1.2.6.8 Possibilité de découverte de service :

Les services sont complétés par des métadonnées communicatives par lesquelles ils peuvent être efficacement découverts et interprétés.

Cette méthode diminue la redondance des services, augmente la flexibilité et renvoi une information très rapide.

1.2.7 Approches de SOA

Deux approches sont définies pour implémenter l'architecture SOA tel qu'il est présenté dans la figure***1.4.***

1.2.7.1 Bottom-Up :

Cette approche est inventée par Krafzig and Slama [07] **, Son** principe est de commencer par un petit groupe et on jouter jusqu'à on obtient une grande entreprise.

1.2.7.2 Top-down :

Cette approche aussi est inventée par Krafzig and Slama. On peut dire que c'est l'adverse de Bottom-Up.

Top-down est la décomposition d'un système ou problème jusqu'à l'obtient d'un petit service de base.

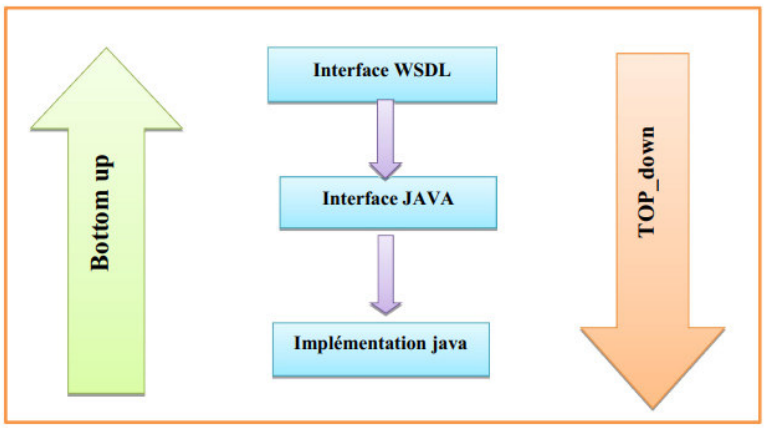

Figure 1.3 : «Approche de développement SOA»

1.2.8 Les Avantages de SOA

La mise en œuvre d'une approche SOA au sein d'une infrastructure informatique fournit des avantages considérables, parmi ces avantages [07]:

- **Développement** : les développeurs se concentrent sur des fonctions élémentaires Sans se préoccuper de l'infrastructure d'accueil.
- **Optimisation de la qualité des services :** la technologie SOA favorise considérablement la réutilisation accrue des modèles et des services, laquelle se traduit par une amélioration de la fiabilité et de la qualité des services par le biais de multiple cycle de test effectués sue les modèles par différents spécialistes.
- **Modification en temps réel d'une simulation :** une simulation peut, en effet être, modifiée

  Sur les processus métiers eux-mêmes. Le travail du développeur est simplifié et plus sécurisé
- **La réutilisation des modèles :** ouvre naturellement les services à une consommation par des applications externes.
- **L'indépendance des fonctions :** par rapport aux processus métiers et à l'architecture d'accueil, la SOA préserve l'évolutivité, la flexibilité et l'intégration.
- **Architecture interopérable** : indépendance entre les services, indépendance de plateforme, langage de programmation, système d'exploitation.
- **Réduction des coûts de développement** : Les services SOA sont aisément réutilisables et peuvent être rapidement assemblés, tels des composants sous la forme de nouvelles applications composites sans nécessiter pour autant une programmation manuelle coûteuse.

- **Allégement des coûts de maintenance** : Les services réutilisables diminuent le nombre et la complexité interne des services informatiques, réduisant ainsi la charge de travail de maintenance requise pour l'intégralité de la gamme de services. Par une amélioration de la fiabilité et de la qualité des services par le biais de multiples cycles de tests effectués par différents utilisateurs.
- **Diminution des coûts d'intégration** : Les services normalisés fonctionnent parfaitement les uns avec les autres, garantissant ainsi la connexion rapide et aisée d'applications disparates.
- **Minimisation des risques** : Des services moins nombreux et réutilisables contribuent à diminuer les risques globaux de non-conformité des opérations.
- **Adaptabilité aux changements** : La possibilité de mettre en place de nouveaux processus métier ou par la composition des processus métier existants facilite la mise en œuvre de solution répondant à un nouveau besoin métier.
- **Assurer un alignement entre le métier et le TI** : SOA permet d'introduire plusieurs niveaux d'abstraction. En effet, elle permet d'introduire une nouvelle couche intitulée service qui encapsule les fonctionnalités techniques et permet une meilleure communication entre le métier et le système TI. Une telle approche facilite la traduction des représentations logiques du métier (processus et entités métiers) en représentation physique (les services).
- **Accroître l'agilité de l'organisation** : cette agilité est mesurée par l'efficacité par laquelle une entreprise peut répondre au changement (réactivité).
- **Vers plus de sécurité** : Le système de sécurité est basé au niveau du service .c'est le point central qui permet de renforcer la sécurité (multi niveaux d'authentification, authentification)

1.2.9 Limites de l'architecture SOA

Malgré les avantages cités ci-dessus, l'adoption d'une architecture orientée services peut poser de sérieux problèmes et entraîner de nouveaux défis, auxquels il faut faire attention.

Ces problèmes ou limites seront détaillés dans la section actuelle. [07]**,** Un sondage réalisé en 2006 sur 85 entreprises (Plouin et al, 2007) (Figure 1.4), montre que :

Les principales limites d'une architecture orientée services sont :

• L'aspect sécurité qui est à surveiller surtout dans le cas où les services sont exposés via le Web.

• Les outils de développement utilisés pour la mise en œuvre d'une SOA expriment souvent un manque de productivité.

Les plateformes d'intégration SOA souffre d'un manque considérable de robustesse.

Comme toute technologie, SOA a ses avantages et ses limites, l'important est de bien profiter de ses apports et savoir pallier ses limites, pour cela il faut adopter une stratégie bien déterminée et savoir choisir la manière avec laquelle on va mettre en place cette architecture.

Figure 1.4 : «Les principales Limites de l'architecture SOA tirés de plouin (2007)»

1.3. Les services Web

Les Services Web ont été conçus pour répondre en premier lieu à l'exigence d'interopérabilité. En fait, les acteurs du monde informatique désiraient supporter l'interopérabilité de façon uniforme et extensible. Il fallait concevoir des solutions qui puissent être intégrées et/ou utilisées par les standards HTTP et HTML. Les Services Web sont le résultat de la convergence de différentes initiatives basées sur XML comme langage de représentation de l'information.

Les Services Web sont basés sur des technologies standardisées, ce qui réduit l'hétérogénéité et fournit un support pour l'intégration d'applications. De plus, les Services Web sont le support pour de nouveaux paradigmes, comme le traitement et l'architecture orientée service.

Les Services Web ont donc plusieurs avantages comme l'utilisation de standards universels, l'indépendance de la plate-forme, un environnement universel pour les systèmes d'information distribués, l'utilisation de plusieurs protocoles de transfert (par exemple HTTP, SMTP et FTP), le codage des messages en XML, un comportement compatible avec les pare-feu, une facilité d'adaptation aux systèmes plus anciens (Legacy Systems), et la localisation par URI (Uniform Resource Identification). [07]

1.3.1 Définitions

Depuis les dernières années, l'utilisation de Services web a connu une popularité grandissante. Ces services sont très utilisés notamment par les entreprises pour rendre accessible leurs métiers ou leurs données via le web. Les services web ont été proposés initialement par IBM et Microsoft, puis en partie standardisés par le W3C (le consortium du World Wide Web). Plusieurs définitions des services web ont été proposées dans la littérature.

D'un point de vue technique, un service Web est une entité logicielle offrant une ou plusieurs fonctionnalités allant des plus simples aux plus complexes. Ces entités sont publiées, découvertes et invoquées à travers le web grâce à l'utilisation d'internet comme infrastructure de communication ainsi que de formats de données standardisés comme XML.

Cette définition met en évidence uniquement le standard XML, donc n'importe quelle application orientée Internet qui garantit ces caractéristiques et qui utilise les technologies basées sur XML est aussi un Service web.

IBM1 : « Les services web sont la nouvelle vague des applications web. Ce sont des applications modulaires, auto-contenues et auto-descriptives qui peuvent être publiées, localisées et invoquées depuis le web. Les services web effectuent des actions allant de simples requêtes à des processus métiers complexes. Une fois qu'un service web est déployé, d'autres applications (y compris des services web) peuvent le découvrir et l'invoquer ».

Les deux premières définitions affirment que les services web sont accessibles par d'autres à travers le web en utilisant des protocoles et des formats standards, mais elles ne mettent pas en évidence les technologies utilisées pour mettre en œuvre un service web.

W3C : « *Un service web est un système logiciel identifié par une URI et conçu pour supporter l'interaction interopérable de machine à machine sur un réseau. Il possède une interface décrite dans un format exploitable par la machine, c.à.d. décrite en WSDL (web Services Description Language). D'autres systèmes interagissent avec le service web d'une façon prescrite par sa description en utilisant des messages SOAP (Simple Object Access Protocol), typiquement en utilisant HTTP (HyperText Transfer Protocol) avec une sérialisation XML en même temps que d'autres normes du web »*

Cette définition met l'accent sur les standards de l'Internet et l'interface ouverte qui permet les invocations des services. Cependant, elle n'est pas encore suffisamment précise parce qu'elle ne mentionne pas la découverte de services web (UDDI)

Le dictionnaire Webopedia3 : Une définition plus raffinée des services web est fournie par le dictionnaire Webopedia3 qui définit un service web comme « une manière standardisée d'intégration des applications basées sur le Web en utilisant les standards ouverts XML, SOAP, WSDL, UDDI et les protocoles de transport de l'Internet. XML est utilisé pour représenter les données, SOAP pour transporter les données, WSDL pour décrire les services disponibles, et UDDI pour lister les fournisseurs de services et les services disponibles » [08].

1.3.2 Caractéristiques des services Web

- Les services Web peuvent être définis, décrits et découverts à travers le Web qui permet une accessibilité facile.
- Un service Web est accessible vie le réseau à partir de leur identification URL.
- Un service Web est fonctionnel quel que soit la plateforme, il est complètement indépendant des langages de programmation et des systèmes d'exploitation.

- Les services Web sont décrits leur interface par des langages standards basés sur XML.
- Les services Web puissent peuvent communiques par des différentes applications et invoquées à distance à travers l'utilisation des protocoles comme (HTTP, SMTP…) pour échanger les données.
- Les services web peuvent être communiqués entre eux facilement grâce à utilisation le langage XML.
- Une habilité à interagir de manière synchrone et asynchrone. [09]

1.3.3 Les applications des services Web

Les technologies des services Web peuvent être appliquées à toutes sortes d'applications auxquelles elles offrent des avantages considérables en comparaison aux anciennes API propriétaires, aux implémentations spécifiques à une plate-forme et à quelques autres restrictions classiques que l'on peut rencontrer (multiplateforme, multi-langage, disponible sur Internet avec une information actualisée disponible en temps réel, ...).

L'application des services Web est multiple, autant dans les domaines du **B2C, B2B** que pour des domaines de gestion, par exemple gestion de stock, gestion commerciale, etc...

- B2C (Business to Consumer) : Qualifie une application, un site Internet destiné au grand public.

- B2B (Business to Business) : Qualifie une application, un site Internet destiné au commerce de professionnel à professionnel.

Les services Web peuvent être utiles dans la plupart des scénarios applicatifs lorsque la communication peut être établie sur un modèle bidirectionnel (requête/réponse). C'est néanmoins loin d'être aussi limitatif, beaucoup d'autres modèles peuvent avoir recours aux services Web, sans même que vous vous en rendiez compte. Les entreprises qui mettent à disposition leurs services Web permettent aux développeurs intéressés par ses fonctionnalités de les réutiliser sans avoir à les recoder. Le principe des services Web permet d'avoir un partage des fonctionnalités et facilite grandement le développement. [10]

1.3.4 Fonctionnement des services web

Scénario 1 : services web non publics [11]

- Services web non publiés dans un annuaire UDDI,
- Services web dont le point d'accès est connu des utilisateurs du service,

- Généralement des SW intranet, des SW de type B2B (Business to Business), …

Figure 1.5 : « Fonctionnement d'un SW à accès non public »

Scénario 1 : services web publics

- Services web publiés dans un annuaire UDDI,
- Généralement des SW de type B2C (Business to Consumer), ex : agence de voyage, …
- Le SW et l'annuaire UDDI qui le publie peuvent ne pas résider sur la même machine.

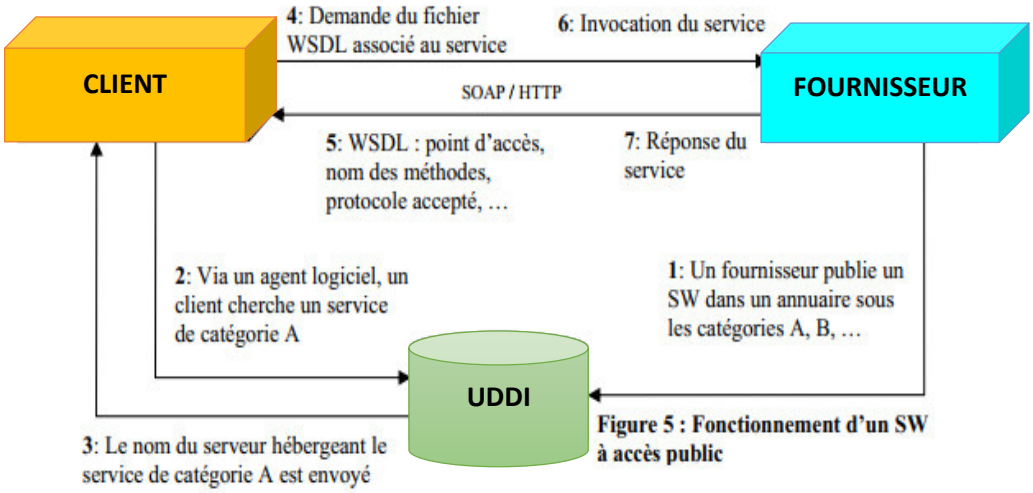

Figure 1.6 : « Fonctionnement d'un SW à accès public »

1.3.5 Architecture des services Web

L'architecture des services Web est une architecture orientée composant (SOA Service Oriented Architecture). L'architecture SOA est un modèle qui définit un système par un ensemble de composants logiciels distribués, les composants dans un système distribué n'opèrent pas dans un même environnement de traitement et sont obligés de communiquer par échanges de messages afin de solliciter des services dans le but d'accomplir le résultat souhaité. La définition de l'architecture des services Web consiste à mettre en évidence les concepts, ainsi que l'ensemble des relations et des contraintes entre ces concepts. Les principaux concepts intervenant dans l'architecture des services Web sont cités dans la table 2.1 : [12]

La mise en œuvre d'une application basée sur l'AOS repose sur un cycle de vie composé de trois phases Préliminaire, Modélisation et Assemblage, la figure 1.7 l'illustre ;

Concepts	Définitions
le fournisseur du service	désigné le serveur qui héberge les services déployés
le client du service	représente l'application cliente qui invoque le service
le service	désigné les fonctionnalités d'un agent logiciel qui implémente le service
la description du service	c'est la spécification du service exprimée dans un langage de description interprétable par les machines, c'est-à-dire une description technique dans laquelle le service est vu en termes de messages, de types, de protocoles de communication et d'une adresse physique
les messages	C'est la plus petite unité d'échange entre les clients et les services .La structure des messages qui permettent l'invocation d'un service doit être exprimée dans la description du service
la ressource	désigne l'identifiant du service ; c'est-à-dire son URI (Uniform Resource Identifier)

Table 1.1 : «Ensemble de concepts et leurs définitions»

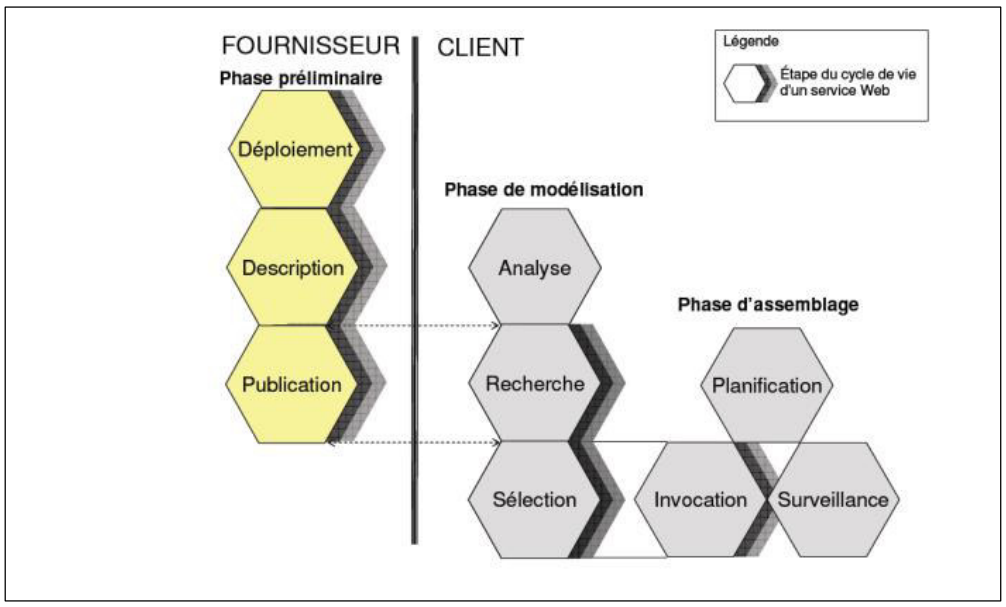

Figure 1.7 : «Illustration du cycle de vie d'une application à base d'architecture orientée service»

La première phase du cycle de vie est la phase préliminaire réalisée par un fournisseur de services élémentaires. Cette phase est composée de trois étapes : [12]

- **Le déploiement** : est spécifique à chaque fournisseur et correspond à l'implémentation du service qui, par la suite, sera décrit et publié
- **La description** : Afin que le service Web déployé puisse être visible, dans le but d'être utilisé (et réutilisé), le fournisseur doit le décrire à l'aide du standard WSDL
- **La publication** : La recherche des services en vue de les utiliser (et réutiliser) repose sur leur publication préalable dans un registre. UDDI (Universal Description Discovery, and Integration) [Clement et al., 2004] avait à l'origine l'ambition de devenir le registre universel de services Web.

Les deux phases suivantes du cycle de vie sont réalisées par le client (le concepteur d'applications à base d'AOS). La phase de modélisation est composée de trois étapes :

- **L'analyse** des besoins préalable à la conception et se situe au niveau interne des organisations.
- **La recherche** : La recherche peut se faire via les registres dans lesquels les fournisseurs ont publié leurs services Web.

- **La sélection** : Le client doit ensuite sélectionner parmi la collection de services Web issus de l'étape de recherche, le service Web qui convient le mieux à ses attentes.

La dernière phase du cycle de vie est la phase d'assemblage. Cette phase est composée de trois étapes :

- **La planification** : lors de cette étape, le client définit la composition de services Web.
- **L'invocation** : lorsque le client connaît au préalable le service Web qu'il veut invoquer (par l'intermédiaire des étapes précédentes telles que la recherche et la sélection), son invocation peut être réalisée via le protocole de communication standard SOAP.
- **La surveillance** : l'exécution d'une composition de services Web engendre des problèmes tels que l'indisponibilité d'un service Web. Lorsqu'un service Web faisant partie d'une composition est indisponible, l'étape de surveillance doit reposer sur des mécanismes de compensation afin de sélectionner un nouveau service Web répondant au mieux aux objectifs du service Web défaillant.

1.3.6 Principales technologie de développement de service web.

L'architecture fonctionnelle des services web que nous avons présenté précédemment montre l'utilisation de nombreux langages et protocoles durant le déploiement et l'invocation des services web. L'atout principal des protocoles SOAP et UDDI ainsi que du langage WSDL est de se reposer sur le langage XML (eXtensible Markup Language). Cette même architecture montre aussi que les services web se basent sur l'utilisation des normes actuelles d'Internet comme le protocole HTTP. [08]

Dans cette section, nous allons présenter d'abord le langage XML. Après quoi nous définirons les déférentes couches horizontales illustrées dans la figure 1.8 qui décrit la pile de langages, de protocoles et de modèles des services web.

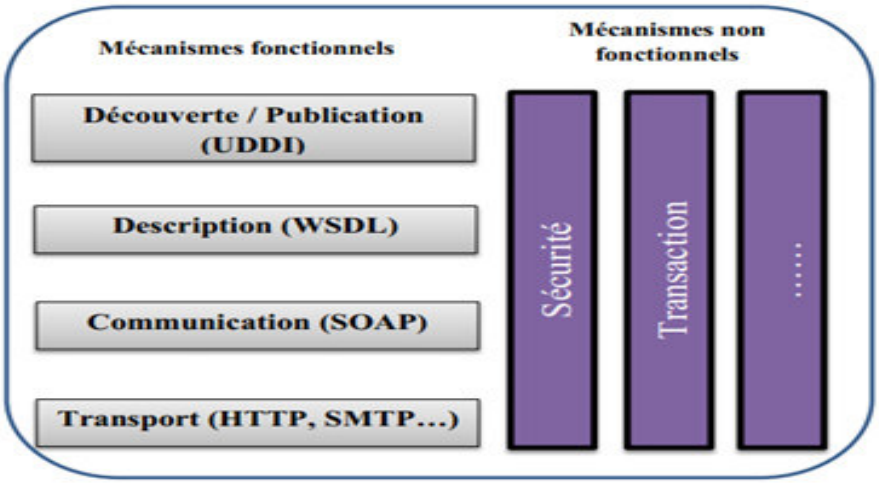

Figure 1.8 : « La pile protocolaire simplifiée des services Web »

1.3.6.1 Le langage XML (eXtensible Markup Language)

Le langage XML standardisé par le W3C en 1998 est aujourd'hui largement reconnu et utilisé par de nombreuses entreprises comme format universel d'échange de données. XML est un métalangage de représentation de données. Il définit un ensemble de règles de formatage pour composer des données valides. XML constitue la technologie de base des architectures Web services ; c'est un facteur important pour contourner les barrières techniques. XML est un standard qui permet de décrire des documents structurés transportables sur les protocoles d'Internet.

En effet, il apporte à l'architecture des services web l'extensibilité et la neutralité vis à vis des plateformes et des langages de développement De plus, grâce à la structuration, XML permet la distinction entre les données des applications et les données des protocoles.

La technologie des services web a été conçue pour fonctionner dans des environnements totalement hétérogènes. Cependant, l'interopérabilité entre les systèmes hétérogènes demande des mécanismes puissants de correspondance et de gestion des types de données des messages entre les différents participants (clients et fournisseurs). C'est une tâche où les schémas de type de données XML s'avèrent bien adaptés. C'est pour cette raison que la technologie des services web est essentiellement basée sur XML ainsi que les différentes spécifications qui tournent autour (les espaces de nom, les schémas XML, et les schémas de Type) [08].

Parmi les spécifications XML, nous soulignons :

- ✓ **XSD (XML Schema)** : c'est un langage qui sert à décrire formellement un vocabulaire .
- ✓ **XSLT (Extensible Stylesheet Language Transformations)** : est utilisé pour transformer un document XML basé sur un certain schéma en un autre document XML qui peut être un document lui-même basé sur un autre schéma.
- ✓ **XPath (XML Path Language)** : fournit une syntaxe d'expressions utilisées pour créer des chemins de localisation.

```
<personne>
   <nom>
         <prenom>Houssem</prenom>
         <nomdefamille>Chemmar</nomdefamille>
   </nom>
   <adresse>
         <numero>28</numero>
         <rue> rue des Lakhdir</rue>
         <ville>Biskra</ville>
         <pays>Algérie</pays>
   </adresse>
</personne>
```

Figure 1.9 : «*Exemple de document XML*»

1.3.6.2 La couche de Transport HTTP (Hypertext Transfer Protocol)

Cette couche s'intéresse aux protocoles de transport de bas niveau, ces derniers vont transporter les requêtes et les réponses échangées entre services. HTTPS est la variante sécurisée de ce protocole. Le protocole le plus utilisé et recommandé par le consortium WS-I (Web Service Interoperability) est l'HTTP, mais d'autres implémentations peuvent utiliser un autre ensemble de protocoles tels que : FTP, JMS (java messagerie services), SMTP, etc.

Le protocole HTTP, à travers les Cookies, fournit un moyen pour "marquer" l'agent client de façon suffisamment précise :

- L'application est marquée, puisque c'est dans le container de cette application que l'information de marquage est stockée.
- L'utilisateur d'une machine multi-utilisateur est marqué, puisque les informations de marquage sont stockées dans le profil particulier de cet utilisateur. [13].

- **Structure Du HTTP :**

✓ **Notion d'entité :** Le HTTP est un protocole destiné à transporter des entités document. Il est constitué de messages-requêtes (du client au serveur) et de messages-réponse (du serveur au client). Il implémente donc parfaitement un paradigme client-serveur en ce sens que :

- Le client (agent utilisateur) est toujours l'initiateur de la requête (PULL).
- Le serveur est toujours en attente d'un client.

Les messages transportent des entités. Le cas le plus fréquent est celui d'une entité unique. L'entité est toujours définie par :

- Un entête d'entité (méta informations sur le contenu et la transmission).
- Un corps d'entité (le contenu).

Le protocole admet cependant que plusieurs entités puissent être transportées par le même message. C'est le cas du téléchargement de fichiers accompagnant des données de formulaire, ou de téléchargements multiples. (Format multipart). Dans ce cas, le message doit permettre de séparer proprement les différentes entités qu'il transporte.

- **Construction De La Requête**

✓ **Le libellé de requête :**

La requête HTTP, comme son nom l'indique est une demande à un serveur pour qu'il exécute un ordre (principalement d'obtenir un document). Ces ordres sont symbolisés par des verbes.

Les verbes les plus connus sont :

- **HEAD** : "donne-moi les métas informations concernant un document" ;
- **GET** : "donne-moi le contenu du document (et ses méta informations par la même occasion)" ;
- **POST** : "fais ce que tu veux (ou tu peux avec les données que je t'envoie)" ;
- **PUT** : "mets le document que je t'envoie où je te le dis".

✓ **La Cible :**

Elle est représentée par une URL absolue ou relative. Le serveur est chargé de "consommer" cette cible, la traduire en une réalité physique tangible (un fichier physique, un exécutable, un répertoire).

La requête peut informer le serveur d'un certain nombre de prédispositions de l'agent utilisateur. Des mécanismes de gestion de préférences, convenues entre le client et le serveur permettent d'améliorer le service rendu, sans faire appel à des choix explicites de l'utilisateur. Ces préférences sont, par exemple :

Le choix de la langue la plus appropriée.

- Le choix d'un encodage acceptable du côté du client.
- Le choix d'une version suffisamment récente, ou au contraire de versions plus anciennes.
- Les champs de requête sont placés à la suite de la "ligne VERBE" sous la syntaxe :

NomChamp : [ValeurChamp] LF

L'en-tête doit se terminer par une ligne vide, pour indiquer que ce qui suit est un corps d'entité (ou que le message est fini).

✓ **Un corps de requête**

Certains verbes (PUT, POST) supposent très fortement une transmission de contenu du client vers le serveur. Ce contenu pouvant être a priori d'une taille quelconque, il s'agit bien d'un corps d'entité.

Dans ce cas, on doit considérer potentiellement une requête comme étant une transmission d'un document à part entière. Le client pousse un document local vers le serveur.

✓ **Détermination de la longueur du corps d'entité**

Il n'existe pas en HTTP de bloc de fin de message identifiable. D'autre part, virtuellement, le corps de l'entité peut contenir n'importe quelle séquence binaire, y compris une séquence qui pourrait se confondre avec un tel bloc. La question de prévoir la fin du corps d'entité doit être réglée dès l'entête.

Toute requête portant un contenu d'une certaine longueur doit informer le serveur de cette longueur. On utilise le champ :

```
Content-Length:
```

➢ **Résumé**

Une requête mono-entité, dans le cas général se construit ainsi :

```
[verbe] [Url] [version HTTP] LF
Content-Length : [longueur du corps]
[Nom Champ]:[Valeur Champ] LF
...
LF (sépare l'en-tête et le corps d'entité)
[Octets de l'entité]
```

Construction De La Réponse

De cette dernière conclusion, on déduit que le schéma de réponse est symétrique à celui de la requête, à la différence près que le corps d'entité est le plus souvent non vide.

1.3.6.3 La couche de Communication SOAP (Simple Object Access Protocol)

Cette couche spécifie les protocoles d'échanges de documents XML entre le service web et ses clients, elle caractérise aussi le mode d'échange (s'il est bloquant ou non). Le protocole SOAP est adopté comme un standard pour la messagerie entre les services web.

SOAP (Simple Object Access Protocol) [08] est un protocole d'échange de message indépendant des plateformes, c'est un produit de Microsoft et IBM.

La première version de SOAP a été acceptée par le W3C (Word Wide Web Consortium) en 2000. SOAP est constitué de deux parties : une enveloppe XML, et un entête d'un protocole de transport. La spécification du protocole SOAP ne donne aucune indication sur le mécanisme de transport du message.

SOAP fait une séparation entre le message (C.à.d. le document XML) et le moyen de transport utilisé. Actuellement, SOAP utilise des protocoles tels que : HTTP ou SMT pour assurer le transport. L'enveloppe XML contient à son tour trois sous éléments :

- ✓ L'entête « Header » : est optionnel, il peut contenir des informations de sécurité (telles que les signatures électroniques), des informations transactionnelles, des informations de traçabilités, etc.
- ✓ L'élément « Body » : est obligatoire, il contient les éléments suivants :

● Soit le nom de méthode, avec les données correspondantes, ou un simple document XML, pour le cas d'une requête.

● Soit les valeurs de retour pour le cas d'une réponse.

✓ L'élément « Fault » : est optionnel, il fournit des informations sur d'éventuelles erreurs survenues lors de l'analyse du message.

La figure 1.10 représente le format standard d'un message SOAP

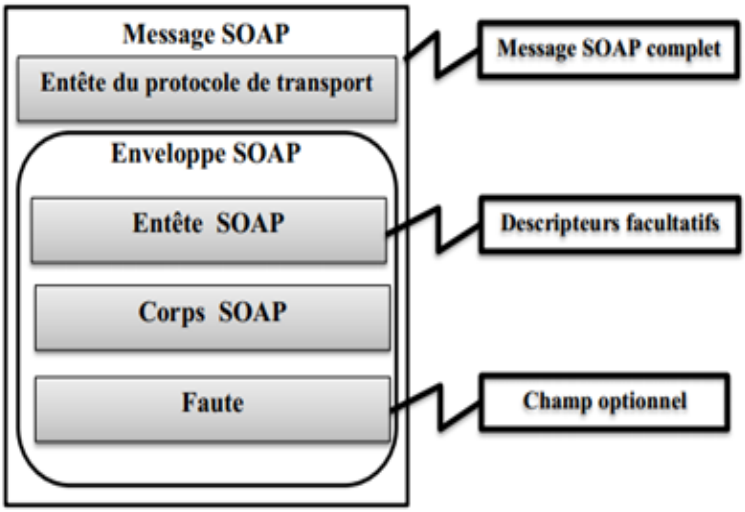

Figure 1.10 : «Schéma d'un message SOAP»

1.3.6.3.1 Protocole SOAP

Les communications entre les différentes entités impliquées [01] dans le dialogue avec le service web se font par l'intermédiaire du protocole **SOAP** (Simple Object Access Protocol).

Ce protocole est normalisé par le W3C , Le protocole SOAP est une surcouche de la couche application du modèle OSI des réseaux. Le protocole applicatif le plus utilisé pour transmettre les messages SOAP est HTTP, mais il est également possible d'utiliser les protocoles SMTP ou FTP, la norme n'impose pas de choix.

Le choix de l'utilisation de protocoles applicatifs, comme par exemple HTTP, est lié aux problèmes d'interconnexion connus des réseaux. Le but des services web étant d'être réutilisables, après publication, à travers tout le réseau Internet, il faut alors donner les moyens de passer les protections telles que les pare-feu (firewalls). Ces derniers autorisent généralement sans aucune restriction le trafic sur les ports liés aux protocoles tels que HTTP,

permettant ainsi le passage sans problème à travers les différents réseaux des messages générés par l'utilisation du protocole SOAP.

1.3.6.3.2 Structure d'un message SOAP

La technologie des services web repose principalement sur le protocole SOAP qui est indépendant des langages de programmation ou des systèmes d'exploitation. Le standard SOAP définit trois éléments composants un message : [01]

L'enveloppe (Enveloppe), l'en-tête du message (Header) et le corps du message (Body) (voir Figure 1.11)

Figure 1.11 : « Structure d'un message SOAP »

- ✓ **Enveloppe SOAP**

L'enveloppe SOAP est l'élément obligatoire d'un message SOAP qui englobe les deux autres parties de ce message (Header et Body). Elle contient le nom du message et le nom du domaine, et permet de préciser la version de SOAP utilisée.

- ✓ **L'En-tête SOAP**

L'en-tête est un élément facultatif dans un message SOAP, marqué par la balise <Header>. Il peut avoir plusieurs fils. Ces fils sont utilisés pour ajouter des fonctionnalités au message SOAP comme l'authentification et la gestion des transactions.

L'en-tête peut utiliser les attributs mustUnderstand et/ou SOAP actor pour déterminer comment le destinataire d'un message doit traiter ce dernier.

- L'attribut acteur de SOAP peut être utilisé pour indiquer le destinataire d'un élément d'en-tête. La valeur de l'attribut d'acteur de SOAP est un URI.
- L'attribut mustUnderstand peut être utilisé pour indiquer si une entrée d'en-tête est obligatoire ou facultative pour être traitée par le destinataire. La valeur de l'attribut de mustUnderstand est "1" ou "0". L'absence de cet attribut est sémantiquement équivalente à sa présence avec la valeur "0".

✓ **Le corps SOAP :**

Le corps du message SOAP est obligatoire, marqué par la balise <Body>. Il permet de transmettre les requêtes et les réponses entre les systèmes, il est composé d'un ou plusieurs sous éléments, qui sont : [01]

- **L'élément FAULT** : il permet d'indiquer les défaillances de transmission des messages SOAP. Il renvoie des informations sur le type d'erreur, une description de l'erreur et l'adresse du serveur SOAP qui a généré l'erreur.
- **L'élément MESSAGE** : il contient les données à transmettre via le protocole SOAP.

1.3.6.3.3 Modèle d'échange de messages en SOAP

Les messages SOAP sont des transmissions fondamentalement à sens unique d'un expéditeur à un récepteur. Lorsqu'une transmission d'un message commence (ex. invocation d'un service web), un message SOAP (ou document XML) est généré. Ce message est envoyé à partir d'une entité appelé le SOAP Sender, localisé dans un SOAP nœud. Le message est transmis à zéro ou plusieurs nœuds intermédiaires (SOAP intermédiaires) et le processus fini lorsque le message arrive au SOAP receiver. Le chemin suivi par un message SOAP est nommé message path, [Ricardo, 2004]. La Figure 1.12 montre les éléments qui participent au processus.

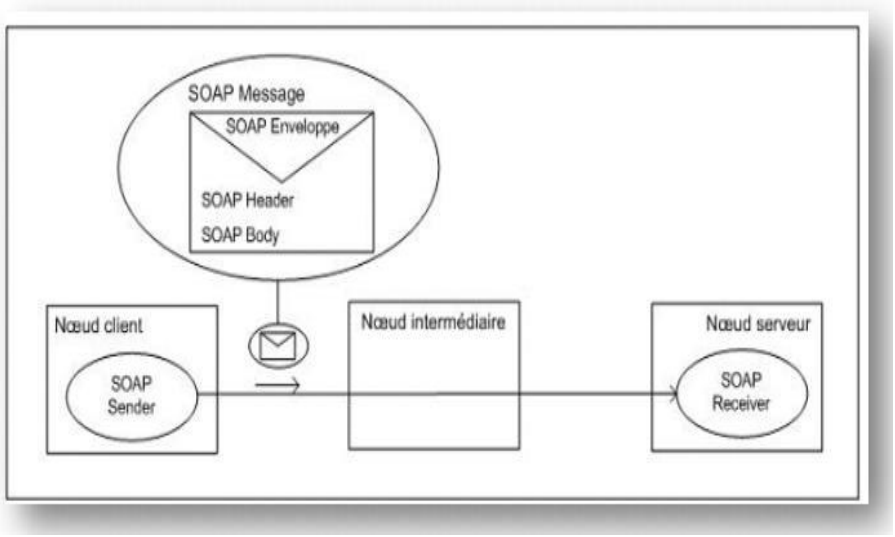

Figure 1.12 : «Modèle d'échange de message en SOAP »

En résumé, le SOAP est un protocole de communication entre les applications fondées principalement sur le protocole HTTP et sur XML, visant à satisfaire un double objectif : [01]

Servir de protocole de communication sur Internet, dans une optique d'intégration d'application, et permettre la communication entre les applications et les services web. Il définit un ensemble de règles pour structurer les messages envoyés. Mais SOAP ne fournit aucune instruction sur la façon d'accéder aux services web. C'est le rôle du langage WSDL.

1.3.6.4 La couche de Description WSDL (Web Service Description Language)

WSDL (Web Service Description Language) **[08]** est un standard du W3C qui permet de définir la structure abstraite et concrète d'un service. C'est un document XML qui décrit la signature des opérations offertes par le service (nom d'opérations, noms et types des paramètres d'entrées/sorties), il définit aussi des liaisons concrètes pour ces opérations telles que le protocole de transport, l'URI du service, le style du service, et les règles d'encodage employées pour les paramètres d'entrées/sorties.

Nous notons que cette interface décrit juste la structure du service (la partie quoi) et non pas le comportement (la partie comment). Le document WSDL comporte six éléments :

<définitions>, <types>, <message>, <portType>, <binding> et <service>. La figure 1.13 offre une représentation concise de la spécification WSDL.

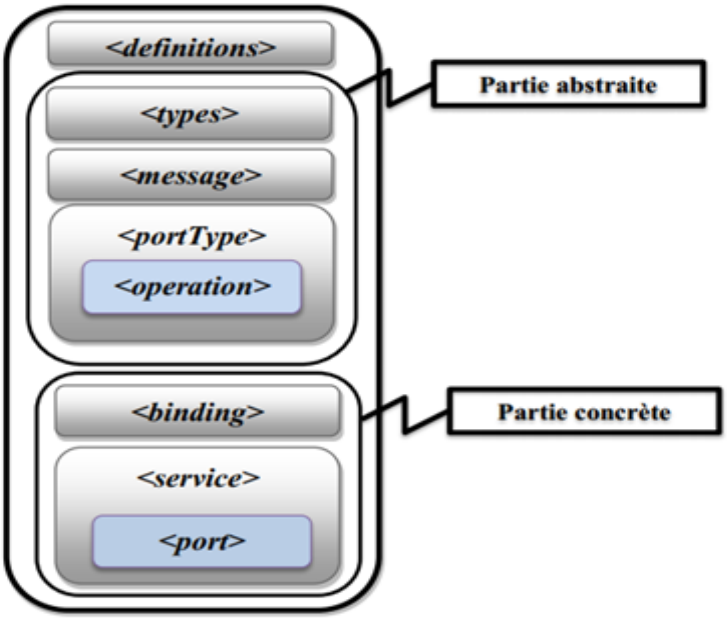

Figure 1.13 : «Structure d'une description WSDL 1.1 »

- ✓ **<définitions>** : L'élément « définitions » représente la racine du document WSDL. Il définit le nom du service web, déclare les différents espaces de nommage utilisés dans le reste du document, et contient tous les éléments qui décrivent le service web.
- ✓ **<types>** : L'élément « types » décrit tous les types de données complexes utilisés entre le client et le serveur. WSDL n'est pas exclusivement lié à un système de typage spécifique, mais il utilise la spécification W3C XML Schéma comme choix par défaut.

Si le service utilise uniquement des types simples, tels que les chaînes de caractères et les entiers, l'élément types n'est pas requis.

- **<message>** : l'élément « message » décrit les messages échangés par le service (les messages entrants et sortants). Il est composé d'un ou plusieurs éléments nommés.
- **<part>** : ces derniers peuvent faire référence à des paramètres d'entrée ou à des valeurs de retour.
- **<portType>** : l'élément « portType » représente une collection d'opérations.
- **<operation>** : est une action abstraite accomplie par le service. Elle peut contenir éventuellement les sous éléments **<input>** et /ou **<output>** décrivant respectivement le message d'entrée et le message de sortie.

La spécification WSDL décrit quatre types d'opérations : [08]

- **Opération à sens unique (One-way)** : Le service reçoit un message. L'opération a donc un seul élément <input>.
- **Opération de type requête – réponse (Request-response) :** Le service reçoit un message et envoie une réponse. L'opération a donc un élément <input>, suivi d'un élément <output>. Pour encapsuler des erreurs, un élément optionnel <fault> peut également être spécifié.
- **Opération de type réponse sollicitée (Solicit-response) :** Le service envoie un message et reçoit une réponse. L'opération a donc un élément <output>, suivi d'un élément <input>. Pour encapsuler des erreurs, un élément optionnel <fault> peut également être spécifié.
- **Opération de type notification (Notification) :** Le service envoie un message.

  L'opération a donc un seul élément <output>.

✓ **<binding>** : L'élément « binding » associe un portType avec des informations concrètes telles que le protocole de transport (http, ftp…), les règles d'encodage de données, et le style du service (RPC ou DOC).

- **Le style RPC** impose un format précis aux messages SOAP (c.à.d. le nom de la méthode et les arguments pour la requête, et la valeur de retour pour la réponse). Ce style est adapté aux échanges synchrones.
- **Le style DOC** autorise n'importe quel document XML bien formé comme requête ou réponse. Ce style est adapté aux échanges asynchrones.

✓ <**service**> : L'élément « service » est constitué d'un ensemble de <port>. Un <**port**> est une association entre un <binding> et un point d'accès (c.à.d. l'url) qui indique l'adresse du service web. Nous pouvons avoir plusieurs ports par services, (un même binding et des URLs différents, ou des « binding » différents et éventuellement des URLs différents).

En plus de ces six principaux éléments, la spécification WSDL définit également les éléments facultatifs suivants :

✓ <**documentation**> : L'élément « documentation » est utilisé pour fournir des informations lisibles par l'utilisateur (par exemple : description textuelle du service en

utilisant le langage naturel). Il peut être inclus à l'intérieur de n'importe quel autre élément WSDL.

- **<import>** : L'élément « import » est utilisé pour importer d'autres documents WSDL ou des schémas XML. Par exemple, deux documents WSDL peuvent importer les mêmes éléments de base et encore inclure leurs propres éléments pour assurer le même service sur deux adresses physiques différentes. Il est à noter que cette fonctionnalité n'est pas supportée par tous les outils WSDL pour l'instant. [08]

Pour plus d'illustration, la figure 1.14 représente la description WSDL du service simple « HelloService ». Ce dernier offre une fonction (opération) appelée « sayHello ». La fonction reçoit un message « SayHelloRequest » avec une chaine de caractères représentant le paramètre d'entrée, et retourne un message « SayHelloResponse » qui contient une autre chaine de caractères. Par exemple, si vous passez le mot «world» comme paramètre, le service renvoie le message d'accueil, « Hello, world ! ».

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="HelloService"
   targetNamespace="http://www.ecerami.com/wsdl/HelloService.wsdl"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://www.ecerami.com/wsdl/HelloService.wsdl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <message name="SayHelloRequest">
      <part name="firstName" type="xsd:string"/>
   </message>
   <message name="SayHelloResponse">
      <part name="greeting" type="xsd:string"/>
   </message>
   <portType name="Hello_PortType">
      <operation name="sayHello">
         <input message="tns:SayHelloRequest"/>
         <output message="tns:SayHelloResponse"/>
      </operation>
   </portType>
   <binding name="Hello_Binding" type="tns:Hello_PortType">
      <soap:binding style="rpc"
         transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="sayHello">
         <soap:operation soapAction="sayHello"/>
         <input>
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="urn:examples:helloservice"
               use="encoded"/>
         </input>
         <output>
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="urn:examples:helloservice"
               use="encoded"/>
         </output>
      </operation>
   </binding>
   <service name="Hello_Service">
      <documentation>WSDL File for HelloService</documentation>
      <port binding="tns:Hello_Binding" name="Hello_Port">
         <soap:address
            location="http://localhost:8080/soap/servlet/rpcrouter"/>
      </port>
   </service>
</definitions>
```

***Figure 1.14 : «Exemple d'un document WSDL»* [08]**

La version 1.0 de l'interface WSDL a été créée en 2000 par IBM, Microsoft et Ariba, ensuite le consortium W3C a publié la version WSDL 1.1 en mars 2001 comme une note.

En juin 2007, la version WSDL 2.0 a été créée comme une recommandation W3C (C.à.d. norme). WSDL 2.0 a apporté quelques changements à la version WSDL 1.1, ils sont résumés comme suit :

- **<portType>** est devenu **<interface>**.
- **<port>** est remplacé par **<endpoints>**.
- **<message>** est supprimé et l'élément **<xs : element>** est utilisé pour spécifier les données échangées. **[28]**

1.3.6.5 La couche de découverte et de publication UDDI (Universal Description Discovery and Integration Registry)

UDDI (Universal Description, Discovery and Integration) est une spécification d'annuaires de services web, cette norme W3C propose un ensemble de structures à publier par les fournisseurs de services. Ces structures sont formalisées en XML, elles proposent 03 types d'informations (voir la figure 1.15) : [08]

- ✓ **Les pages blanches** : noms, adresses, identifiants et contacts des entreprises enregistrées. Ces informations sont décrites dans des entités de type « Business Entity ». Cette description inclut des informations de catégorisation permettant de faire des recherches spécifiques dépendant du métier de l'entreprise.
- ✓ **Les pages jaunes** : donnent les détails sur le métier des entreprises et les services qu'elles proposent. Ces informations sont décrites dans des entités de type « Business Service ».
- ✓ **Les pages vertes** : Les informations techniques sur les services, tels que la façon d'agir avec eux : des définitions de processus métier ; et l'URL du WSDL de service.

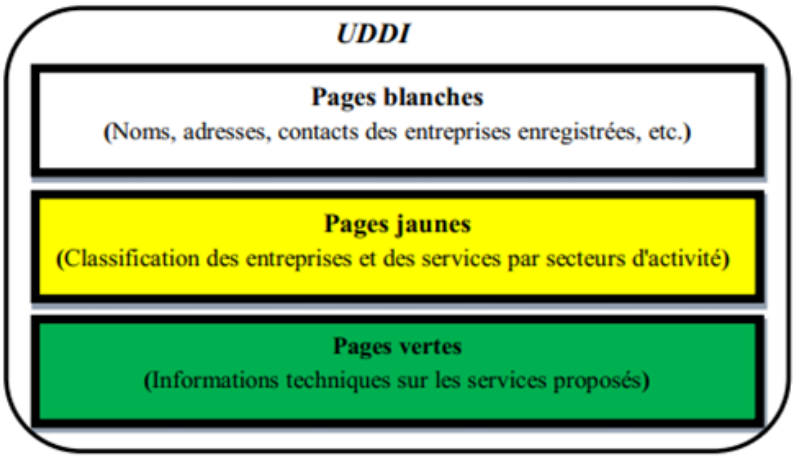

Figure 1.15 : «Le contenu de l'annuaire UDDI »

La norme UDDI offre aussi une API aux applications clientes, pour rechercher des services et/ou ses fournisseurs, ajouter ou modifier des services ou des entreprises. Nous distinguons deux types d'annuaires UDDI (publiques et privés) : [08]

L'annuaire UDDI publique est implémenté sous forme d'un réseau de nœuds UDDI, ces nœuds sont synchronisés, chacun d'eux est possédé par une entreprise donnée (telles que SAP,

IBM et Microsoft). La publication d'un service chez une entreprise propage automatiquement ses informations (pages blanches, jaunes et vertes) aux différents nœuds UDDI. L'accès à l'ensemble des informations des registres peut se faire à n'importe quel nœud UDDI. Ce type d'annuaire est gratuit.

L'annuaire UDDI privé permet à une entreprise de choisir les partenaires pour lesquels elle autorise la publication et l'invocation de ses services web.

Les structures de données d'un UDDI sont illustrées dans la figure 1.16

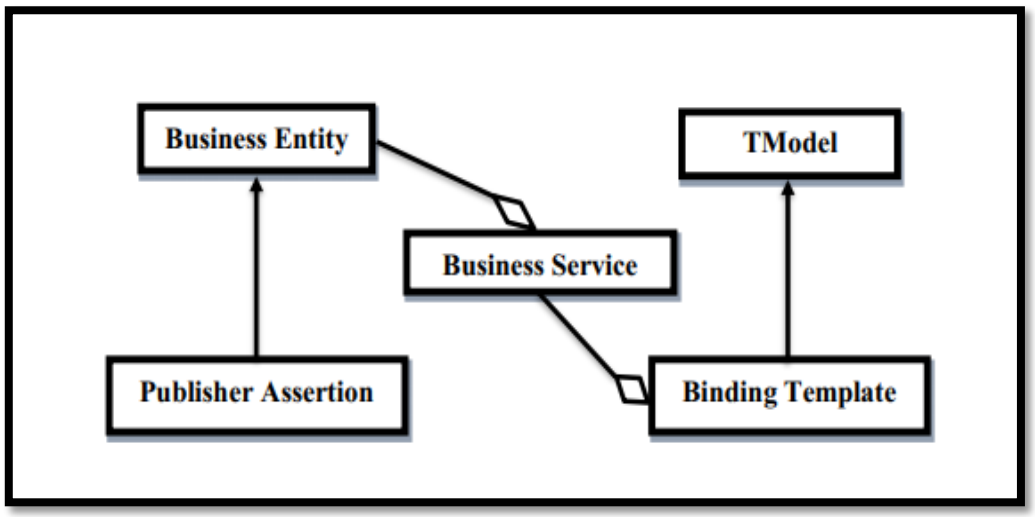

Figure 1.16 : «Les structures de données d'UDDI»

- ✓ **La partie « pages blanches »** : est constituée de deux éléments **<businessEntity>** et **<Publisher Assertion>**.
  - **<businessEntity>** donne des informations de contact sur l'organisation fournissant les services (les noms des dirigeants, les emails, les numéros de téléphones, le site web, etc.).
  - **<Publisher Assertion>** spécifie les liens d'affiliation entre deux entreprises (mère et fille). Lorsque deux éléments <businessEntity> référencent la même <Publisher Assertion>, nous parlons de relation d'affiliation entre ces <businessEntity>.

- ✓ **La partie « pages jaunes »** : est constituée des éléments **<business Service>**.[08]

- **<business Service>** décrit un ensemble de services fournis par une organisation (le nom du service, la description textuelle, les informations de classification).

  Un **<business Service>** est un sous élément de **<businessEntity>**.

✓ **La partie « pages vertes »** : est constituée des éléments **<bindingTemplate>** et **<tModel>**.

- **<bindingTemplate>** spécifie des informations techniques, telles que l'URI du service. Un <bindingTemplate> est un sous élément de **<business Service>**.
- **<tModel>** définit des structures d'informations techniques telles que l'interface WSDL, les taxonomies industrielles, les ontologies, etc. Un élément **<tModel>** peut être réutilisé par plusieurs éléments <bindingTemplate>.

Le protocole d'utilisation de l'UDDI offre trois fonctions de base :

- **Publish** : permet de publier un nouveau service.
- **find** : permet d'interroger l'annuaire UDDI.
- **bind** : permet d'établir une connexion entre l'application cliente et le service.

1.3.7 Pile des services

D'autres spécifications sont aussi nécessaires pour exprimer les aspects critiques et réaliser l'interaction entre des applications métiers conséquentes. Nous citons en vrac : [14]

- Web Services Choreography Description Language (WS-CD) pour la réalisation de chorégraphies de services.
- Web Services Transactions (WS-Transaction) quant à lui permet le support d'interactions de services fiables.
- Web Services Security (WS-Security), pour la description des exigences de sécurité,
- Web Services Reliable Messaging (WS-Reliabl eMessaging) pour la description des messages.

Un résumé de ces différentes spécifications est fourni par le schéma de la figure 1.17.

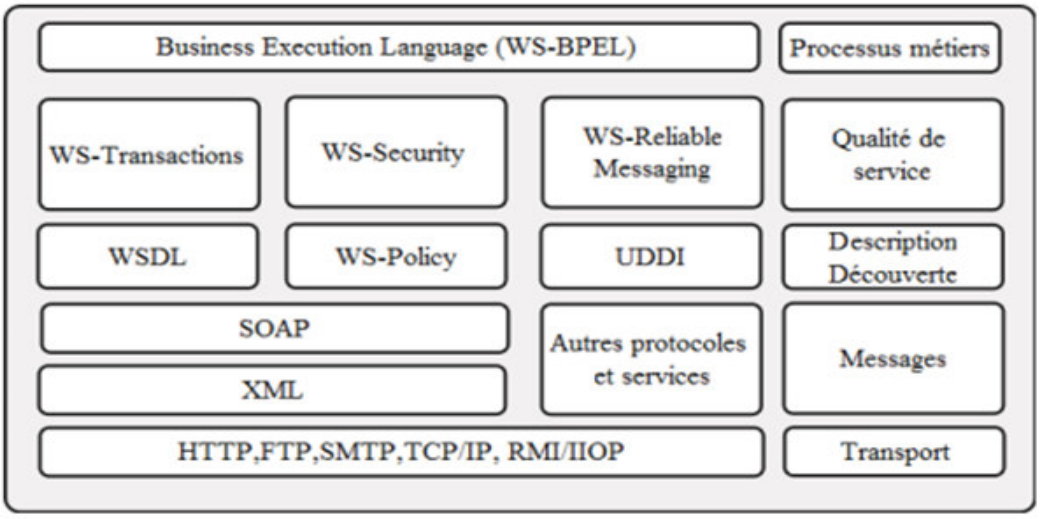

Figure 1.17 : «Pile des services web »

1.3.8 Avantages des Services Web

La technologie des services web est populaire et couramment utilisée car elle offre des avantages intéressants pour les utilisateurs des systèmes distribués : [01]

- Les services web réduisent le temps de mise en marché des services offerts par les diverses entreprises.
- Les services web permettent à des programmes écrits en des langages différents et sur des plateformes différentes de communiquer entre eux par le biais de certaines normes. En d'autres termes, les services web permettent une meilleure interopérabilité entre les systèmes informatiques hétérogènes
- Les services web permettent de tirer profit des infrastructures informatiques existantes et des plateformes ouvertes (Internet, par exemple).
- Les services web permettent d'échanger les informations en toute sécurité à l'aide des protocoles standards comme HTTP qui utilise le port 80 qui fonctionne aux nombreux pare-feu sans nécessiter des changements sur les règles de filtrage.
- Les protocoles et les formats de données sont offerts, le plus possible, en format texte pour que la compréhension du fonctionnement des échanges soit plus intuitive.
- Grâce aux services web, les coûts sont réduits par l'automatisation interne et externe des processus commerciaux.

- Le client doit simplement exprimer ces besoins et les entreprises doivent simplement publier leur service web. [15]
- Les services Web offrent une interface simple : à partir de WSDL, ils ont été créés en fonction de langage XML, qui accorde la relation de compréhension entre le fournisseur et le client, et aussi entre l'entreprise et la société.
- Le faible couplage des interfaces associées aux services web, ainsi que leur architecture modulaire, permettent la réutilisation des services au sein d'autres modules .
- Les outils de développement, s'appuyant sur ces standards, permettent la création automatique de programmes utilisant les services Web existants**.** [03]

1.3.9 Limites des Services Web

La technologie des services web comporte plusieurs inconvénients dont : [12]

- **Problèmes de sécurité** : Il est facile de contourner les mesures de sécurité mises en place par les pare-feu -l'utilisation du protocole HTTP (tel que mentionné ci-haut) n'a pas que des avantages -car les normes de sécurité des services web laissent encore à désirer. CORBA, par exemple, qui est une technologie plus mûre, est plus sécuritaire.
- **Problèmes de performance** : Les services web sont encore relativement faibles par rapport à d'autres approches de l'informatique répartie telles que CORBA ou RMI.
- **Confiance** : Les relations de confiance entre différentes composantes d'un service web sont difficiles à bâtir, puisque parfois ces mêmes composantes ne se connaissent même pas.
- **Syntaxe et sémantique** : On se concentre beaucoup sur comment invoquer des services (syntaxe) et pas assez sur ce que les services web offrent (sémantique).
- **Fiabilité** : Il est difficile de s'assurer de la fiabilité d'un service car on ne peut garantir que ses fournisseurs ainsi que les personnes qui l'invoquent travaillent d'une façon fiable.
- **Disponibilité** : Les services web peuvent bien satisfaire un ou plusieurs besoins du client. Seront-ils pour autant toujours disponibles et utilisables ? Ça reste un défi pour les services web.

1.4 Conclusion

Les services Web regroupent tout un ensemble de technologies bâties sur des standards (SOAP, WSDL, UDDI, XML). Ils permettent de créer des composants logiciels distribués, de décrire leur interface et de les utiliser indépendamment de la plate-forme sur laquelle ils sont implémentés. La recherche dans ce domaine est très soutenue et s'intéresse entre autres à l'adaptation, la composition, l'orchestration, la sécurité et la sémantique des services Web...etc.

Dans le chapitre suivant nous aborderons le concept de l'adaptation des services Web.

2.1 Introduction

La prolifération des ressources sur le Web induit souvent une surcharge cognitive chez l'utilisateur et rend difficile l'accès à l'information et aux services répondant à ses besoins. Ce problème génère un besoin croissant de mise en œuvre d'un processus d'adaptation lors de cet accès. L'étude de ce type de processus est devenue particulièrement populaire dès le début des années 1990.

En général l'objectif des chercheurs dans le domaine des services web est de rendre pertinents les informations et les services fournis par rapport au profil de l'utilisateur (ses préférences, ses connaissances, etc.). Cependant, avec l'émergence des nouvelles technologies qui permettent à l'utilisateur d'accéder à l'information et aux services en utilisant plusieurs types des dispositifs via des types de réseau différents, un nouveau besoin en adaptation est apparu. Ce besoin est la capacité pour l'adaptation de prendre en compte non seulement le profil de l'utilisateur mais aussi son contexte. Ce contexte correspond principalement aux paramètres du dispositif de l'utilisateur, à son réseau, à sa localisation, etc.

Dans ce chapitre, nous allons présenter l'adaptation des services web en détaillant les différents types et les techniques d'adaptation. Mais nous devons commencer par la description de la notion du contexte qui présente la base de l'adaptation.

2.2. Notion de contexte

2.2.1 Définition de la notion de contexte

Au regard de la littérature, il apparaît qu'il n'existe pas qu'une seule définition pour la notion de contexte. En effet, divers domaines de recherche utilisent cette notion en proposant une ou plusieurs définitions. En ce qui concerne les Systèmes d'Information sensibles au contexte, les premiers travaux de définissent le contexte comme étant :

« La localisation de l'utilisateur, les identités et les états des personnes et des objets qui l'entourent.»

Ils définissent donc, le contexte comme les changements de l'environnement physique de l'utilisateur et des ressources de calcul. Afin de préciser le contexte, ils répondent aux questions « On est où ? », « Avec qui ? », « Quelles sont les ressources qu'on utilise ? ». Par ailleurs, *Brown et al*, décrivent le contexte comme : [16]

« L'identité de l'utilisateur, des personnes et objets qui l'entourent, sa localisation géographique, son orientation, la saison, la température où il évolue... ».

Ils limitent le contexte aux éléments de l'environnement de l'utilisateur et introduisent l'heure, la saison, la température, l'identité et la localisation de l'utilisateur.

En fait, les auteurs dans les premiers travaux limitent la définition du contexte à l'utilisateur et son environnement (localisation, l'heure, etc.). Ce qui pousse les travaux les plus récents dans le domaine des systèmes sensibles au contexte à la recherche d'une définition du contexte plus générale et plus claire. Par exemple, Mostéfaoui et al présentent le contexte :

« Comme ce qui entoure le centre d'intérêt de l'individu et qui apporte des informations additionnelles capables d'aider à la compréhension de ce centre d'intérêt. »

Dans cette définition, les auteurs présentent le contexte en se concentrant seulement sur le contexte logique ; ils ne prennent pas en compte le contexte physique comme la machine de l'utilisateur, la localisation de l'utilisateur, etc.

D'autres auteurs ont préféré une approche plus pragmatique en proposant des définitions plus adaptées pour la conception des systèmes sensibles au contexte.

Lemlouma, définit le contexte comme :

« L'ensemble de toutes les informations de l'environnement qui peuvent influencer le processus de l'adaptation et de la transmission du contenu vers l'utilisateur final ».

Selon *Dey*, le contexte couvre : [16]

« Toutes les informations pouvant être utilisées pour caractériser la situation d'une entité. Une entité est une personne, un lieu, ou un objet qui peut être pertinent pour L'interaction entre l'utilisateur et l'application, y compris l'utilisateur et l'application. »

Cette définition concerne particulièrement la conception des systèmes sensibles au contexte, puisqu'elle tient compte de la pertinence des éléments pour l'interaction entre l'utilisateur et l'application.

2.2.2 Modélisation de contexte

Après avoir proposé notre définition du contexte, nous présentons ici la structure du contexte de l'utilisateur et celle du Service Web.

2.2.2.1 Contexte de l'utilisateur

Nous détaillons dans cette section le contexte de l'utilisateur. Ce contexte est constitué des informations concernant spécifiquement l'utilisateur. Dans ce contexte, nous assemblons toutes les informations nécessaires afin de bien présenter la situation

contextuelle de l'utilisateur. La prise en compte de ces informations augmente la possibilité de restituer à l'utilisateur une réponse pertinente, c'est-à-dire améliorer la qualité de réponse retournant à l'utilisateur.

Ce contexte, qui est mis à jour pour chaque requête donnée par l'utilisateur, est composé de quatre profils :

1. **Le profil de l'utilisateur** qui est composé des caractéristiques statiques (nom, prénom, adresse électronique, etc.) et caractéristiques évolutives qui sont définies par son environnement (sa localisation, temps, date de la requête, etc.) et ses préférences (déclarées par l'utilisateur) ;

2. Le profil du dispositif qui présente d'une part, le contexte du matériel (type du dispositif, taille de l'écran, type de processeur, la mémoire libre, etc.), et d'autre part, le contexte du logiciel (type du système d'exploitation, la version du système d'exploitation, le navigateur etc.),

3. Le profil du réseau qui expose des informations sur le type de réseau, ses caractéristiques, etc.,

4. Le profil de session qui présente la connexion par l'utilisateur au système (la durée de connexion, la date de connexion).

☞ Le contexte du service est composé de trois éléments :

- Le profil du service qui est présenté par le nom, la description (location, temps, langage, etc.), Précondition (mémoire libre, etc.) et QoWS (bande passante, temps de réponse, etc.) de service,
- Le profil du dispositif (matériel, logiciel),
- Le profil du réseau (type, caractéristiques, etc.)

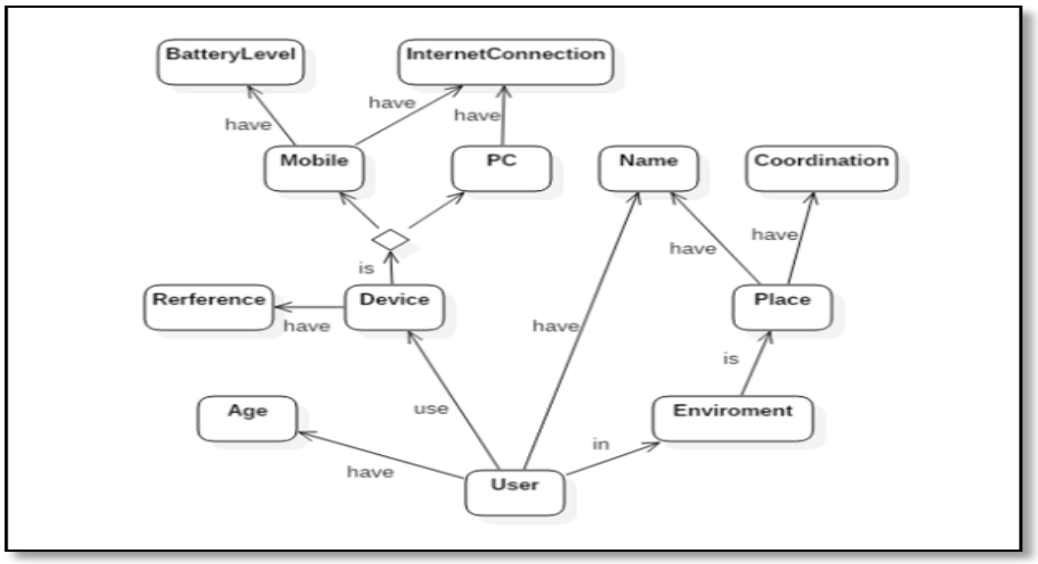

Figure 2.1 : « Exemple de contexte d'utilisateur »

2.2.2.2. Contexte du Service Web

La description d'un Service Web effectuée en WSDL [16] présente seulement des informations sur les fonctionnalités du service telles que les types de données entrantes et sortantes, les opérations réalisées par ce service, etc. Mais, les informations non-fonctionnelles de ce service ne sont pas décrites ; par exemple avec quel type de dispositif nous pouvons configurer ce service, le temps nécessaire pour délivrer à l'utilisateur la réponse finale du service, etc.

Pour cela, nous proposons de prendre en considération le contexte du service. Ce dernier présente des informations supplémentaires sur ce service dans le but de permettre de savoir si ce service est pertinent par rapport au contexte courant de l'utilisateur ou non. Nous regroupons donc, dans ce contexte, les informations non fonctionnelles de ce service.

Ce contexte est composé de trois profils : profil du service, profil du dispositif et profil du réseau.

1. Le profil du service décrit par : [16]

- Le nom du service,
- La description qui contient des informations qui décrivent les données du service.

Comme par exemple la localisation du service, le temps d'ouverture et fermeture du magasin ou du restaurant présenté par le service, le langage d'affichage du service, etc.

- Précondition contient des informations sur des contraintes qui doivent être réalisées/satisfaites avant l'utilisation du service telles que la mémoire libre disponible dans le système hôte, etc.,
- QoWS présente les informations nécessaire pour assurer la qualité du service comme par exemple la bande passante, le temps de réponse, etc.

2. Le profil du dispositif comprend la description du terminal qui doit être employé par l'utilisateur. Il distingue deux éléments :

- **le contexte matériel** : qui comporte les informations telles que le type du dispositif, le type de processeur, la mémoire libre, taille de l'écran, etc., et
- **le contexte logiciel** : Cet élément comporte les informations telles que le type et la version du système d'exploitation, le navigateur, etc.

3. Le profil du réseau comprend des informations sur le réseau auquel est connectée la machine de l'utilisateur tel que le type de réseau, ses caractéristiques, etc.

2.3 Notion de l'adaptation

2.3.1 Définition de l'adaptation

L'adaptation est, selon le grand dictionnaire, l'opération qui consiste à apporter des modifications à un logiciel ou à un système informatique dans le but d'assurer ses fonctions et, si possible, d'améliorer ses performances, dans un environnement d'utilisation. [17]

Dans le domaine de la sensibilité au contexte, l'adaptation est étendue par la notion d'adaptabilité qui caractérise un système qui est capable de changer son comportement lui-même afin d'améliorer ses performances ou de continuer son exécution dans des environnements différents**.**

2.3.2 Pourquoi l'adaptation

Après avoir mis en œuvre une application, plusieurs raisons peuvent conduire à l'adapter.

Ces raisons peuvent être classées en quatre catégories **:** [18]

2.3.2.1 Adaptation correctionnelle

Dans certains cas, nous remarquons que l'application ne se comporte pas correctement ou comme prévu. La solution est d'identifier le module de l'application qui cause le

problème et de le remplacer par une nouvelle version supposée correcte. Cette nouvelle version fournit la même fonctionnalité que l'ancienne, elle se contente simplement de corriger ses défauts.

2.3.2.2 Adaptation adaptative

Même si l'application développée s'exécute correctement, parfois son environnement d'exécution (comme le système d'exploitation), les composants matériels (ou d'autres applications ou données dont elle dépend) changent. Dans ce cas, l'application est adaptée en réponse aux changements affectant son environnement d'exécution.

2.3.2.3 Adaptation évolutive

Au moment du développement de l'application, certaines fonctionnalités ne sont pas prises en compte. Avec l'évolution des besoins de l'utilisateur, l'application doit être étendue avec de nouvelles fonctionnalités. Cette extension peut être réalisée en ajoutant un ou plusieurs modules pour assurer les nouvelles fonctionnalités ou même en modifiant les modules existants pour étendre leurs fonctionnalités tout en gardant l'architecture de l'application.

2.3.2.4 Adaptation perfective

L'objectif de ce type d'adaptation est d'améliorer les performances de l'application. À titre d'exemple, nous nous rendons compte que l'implémentation d'un module n'est pas optimisée. Nous décidons alors de remplacer l'implémentation du module en question. Un autre exemple peut être un module qui reçoit beaucoup de requêtes et qui n'arrive pas à les satisfaire. Pour éviter la dégradation des performances du système, nous diminuons la charge de ce module en installant un autre module qui partage la charge avec le premier. [18]

2.3.3 Classification de l'adaptation

Après avoir présenté la définition de l'adaptation, nous énumérons les différentes classifications utilisées dans les travaux existants d'adaptation. Cette classification élaborée dépend de la manière d'application de l'adaptation : statique ou dynamique, centralisée ou distribuée, comportementale ou architecturale.

2.3.3.1 Adaptation statique

L'adaptation statique ou manuelle consiste à préparer plusieurs versions de la même ressource avant son exploitation. Cette ressource peut être un document, une image, une vidéo ou même une application. La ressource est adaptée manuellement avant de pouvoir la réutiliser dans son nouveau contexte. Cette technique a été adoptée pour plusieurs développements au début de l'apparition des terminaux mobiles afin de pouvoir exploiter des ressources initialement prévues pour des PC standards. Avec une adaptation statique, le résultat est dédié au nouveau contexte d'utilisation de la ressource ce qui garantit la sûreté de son exploitation. Elle présente aussi une solution simple et efficace car chaque fournisseur s'occupe de l'adaptation de la ressource pour ses contextes. Cependant, elle présente l'inconvénient de la difficulté d'évolution pour la prise en charge de nouveaux contextes d'utilisation. [17]

2.3.3.2 Adaptation dynamique

Contrairement à l'approche précédente, l'adaptation dynamique effectue les transformations sur la ressource au cours de son utilisation. Un système d'adaptation automatique intercepte les requêtes des utilisateurs et effectue à la volée des transformations suivant les caractéristiques du contexte d'utilisation actuel. OPERA et MUSA sont des applications où les données sont adaptées dynamiquement aux caractéristiques du terminal utilisé. Elles présentent une adaptation des pages WEB pour différents terminaux en agissant sur leur composition logique, spatiale, temporelle et hypertextuelle. Avec une adaptation dynamique, nous gagnons un temps de développement important puisque la ressource est transformée automatiquement à la demande de l'utilisateur.

Cependant, des problèmes d'adaptation peuvent surgir dans un nouveau contexte d'utilisation inconnu, en fournissant un résultat d'adaptation qui n'est peut-être pas adéquat à ce contexte. En outre, le temps d'attente nécessaire à l'adaptation de la ressource peut gêner l'utilisateur. En effet, l'adaptation dynamique augmente automatiquement la latence de renvoi de l'information demandée par l'utilisateur.

2.3.3.3 Adaptation centralisée

L'adaptation peut concerner une ressource locale à une machine ou distribuée sur plusieurs machines (typiquement un protocole). Dans le cas où la ressource est distribuée, le processus d'adaptation peut être centralisé ou réparti sur les différents pairs où les différentes parties de la ressource ont été stockées. [17]

2.3.3.4 Adaptation distribuée

L'utilisation de cette solution est limitée à des cas particuliers concernant les protocoles de communication**.** Soulève les problèmes de synchronisation de l'adaptation distribuée et propose un protocole dédié pour gérer cette synchronisation.

- **Discussion**

Malgré un grand intérêt pour l'adaptation dynamique, l'adaptation statique reste aussi un moyen sûr et efficace pour assurer l'exploitation des ressources dans différents contextes d'utilisation. Pour assurer une adaptation efficace et ouverte à plusieurs contextes, il est important selon Chaari d'utiliser les deux approches de façon complémentaires [17]

L'adaptation centralisée est plus facile à implémenter que la solution distribuée et elle donne des performances meilleures au niveau du temps d'accès à la ressource adaptée.

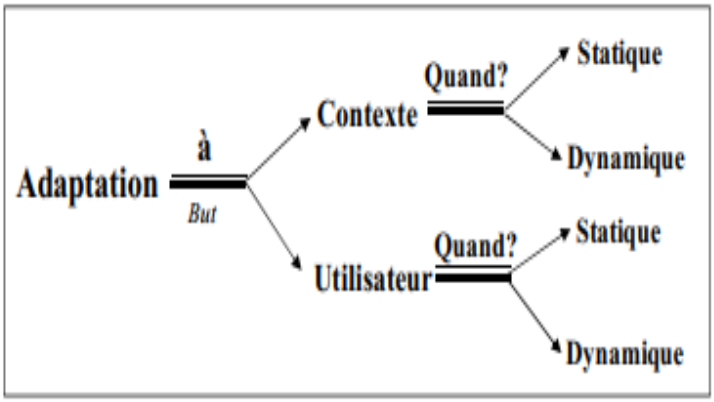

Figure 2.2 : « Classification de l'adaptation en fonction du but » [17]

2.3.4 Paramètres d'adaptation

Les concepteurs des applications doivent concevoir des applications pour être utilisées sur des terminaux qui ont des caractéristiques spécifiques (performances très réduites, petites mémoire vive et morte, taille de l'écran réduite, …). De plus, ces applications doivent utiliser le contexte pour réaliser des activités et fournir l'information appropriée aux utilisateurs sans l'interaction humaine. [17]

Donc, nous devons assurer une adaptation au type du terminal et à l'utilisateur connecté pour garantir une utilisation confortable des applications dans ces nouveaux environnements. Pour réaliser cette adaptation, beaucoup de nouveaux paramètres entrent en jeu :

- ✓ **Paramètres réseau** : dans les réseaux sans fil la bande passante est limitée.
- ✓ **Paramètres de l'utilisateur** : l'utilisateur est devenu le point central de la conception des systèmes d'information. Les concepteurs de ce genre d'application doivent prendre en considération les préférences de l'utilisateur, son emplacement géographique, son profil…
- ✓ **Paramètres du terminal** : la diversité des terminaux mobiles influe sur la conception de ces systèmes d'information. Le comportement de ces systèmes doit s'adapter aux capacités matérielles et logicielles de ces appareils.

Tous ces paramètres forment des contextes d'utilisation différents. Dans la plupart des cas, ces paramètres en plus des changements de l'environnement doivent être pris en compte lors de la conception des applications. L'adaptation des applications comporte :

- ➢ **L'adaptation de l'information** : qui peut porter sur le contenu et la représentation et prendre en compte les différentes informations du contexte.
- ➢ **L'adaptation des services** : qui consiste à modifier leurs comportements pour qu'ils soient compatibles avec le contexte d'utilisation de l'application.

2.3.5 Adaptation des services Web

L'utilisation croissante d'Internet par des utilisateurs ayant des profils variés a par conséquence un besoin croissant d'adaptation. Ce besoin d'adaptation avec le succès des services Web fait émerger les travaux concernant l'adaptation des services Web.

Dans la littérature, les travaux sur l'adaptation s'articulent autour de quatre dimensions, à savoir : [18]

2.3.5.1 L'adaptation du résultat de service Web

Vise à adapter le contenu final du résultat de l'invocation de service Web en ajoutant ou éliminant des données qui sont jugées pertinentes ou non respectivement. La détermination de la pertinence des données est reliée aux informations de contexte telles que les buts de l'utilisateur et ses caractéristiques.

2.3.5.2 L'adaptation de la présentation de service Web

Vise à effectuer des transformations au niveau de l'affichage. En suivant les préférences de l'utilisateur et son contexte, des adaptations sont appliquées sur les caractéristiques visuelles du document présenté à l'utilisateur tel que l'adaptation des caractéristiques graphiques, de la langue de présentation et de l'organisation du document.

2.3.5.3 L'adaptation du comportement

Vise à adapter le comportement des services suivant des changements de l'environnement d'exécution. L'adaptation du comportement du service consiste à changer la logique d'un service. Elle concerne l'occurrence et l'ordre des étapes d'exécution d'un service, ainsi que les relations logiques entre ces différentes étapes : [18]

2.3.6 Le processus d'adaptation

En général, nous pouvons décomposer le processus d'adaptation quelque-soit son type en deux étapes successives : [17]

- La prise de décision concernant les opérations d'adaptation à effectuer,
- L'application des actions d'adaptation en utilisant un mécanisme d'adaptation fourni par le système.

Les conditions qui engendrent l'exécution de chaque étape nous permettent de distinguer deux natures d'adaptation : l'adaptation de nature réactive et l'adaptation de nature proactive.

Si la prise de décision d'adaptation et son exécution, sont conditionnées par la détection d'une situation pertinente, nous disons que c'est une adaptation réactive.

Alors que si la décision d'adaptation est prise au moment de la détection de la situation pertinente alors que son exécution est conditionnée par un autre évènement, nous disons que c'est une adaptation proactive. D'autres types d'adaptations proactives, consistent à prendre la décision d'adaptation avant la détection d'une situation pertinente, simplement en utilisant historique des variations du contexte et un mécanisme d'apprentissage. .

2.3.7 Des approches pour mettre en œuvre l'adaptation

Pour mettre en œuvre le processus d'adaptation et pour la localisation de la fonctionnalité
de ce processus entre le client et le serveur, trois approches générales ont été proposées : côté serveur, côté client et côté proxy.

2.3.7.1 Adaptation côté client

Dans cette approche, le dispositif du client est responsable de la gestion du processus d'adaptation selon les préférences de ce client, les caractéristiques de son dispositif, sa localisation, etc. L'intérêt de cette approche est sa très grande flexibilité : parfaitement au courant des conditions effectives d'utilisation (le contexte), le dispositif du client peut choisir les procédures d'adaptation les plus pertinentes. [16]

2.3.7.2 Adaptation côté proxy

Dans cette approche, c'est le proxy qui est responsable de gérer le processus d'adaptation Un proxy est un nœud intermédiaire placé entre le client et le serveur. Il intercepte la réponse du serveur, effectue l'adaptation en fonction des préférences de l'utilisateur, des capacités des terminaux et de l'état du réseau, etc., puis il envoie le résultat adapté au client. Par exemple, Singh propose d'utiliser les proxys afin de localiser une version des données adaptée. Si aucune version adaptée au contexte n'est trouvée, le proxy s'occupe de transcoder les données afin d'en créer une. , les auteurs intègrent des proxys spécifiques qui sont chargés de réaliser l'interface entre les Services Web et les proxys locaux. **[16]**

2.3.7.3 Adaptation côté serveur

Dans ce cas, le serveur prend la responsabilité du processus d'adaptation ; les fonctionnalités du serveur traditionnel sont étendues par les mécanismes d'adaptation. Par exemple, .les auteurs proposent des techniques d'adaptation pour les documents multimédias échangés sur Internet. De son côté, Mogul s'est focalisé sur la transcription d'images pour le Web. Dans cette approche, l'adaptation est réalisée par deux méthodes : une approche statique et une approche dynamique. [16]

Dans le cas statique, le serveur, pour répondre à la diversité des besoins, stocke plusieurs versions du même document (ex : français/anglais). Les avantages de cette méthode sont : l'efficacité en termes de temps de réponse, le coût processeur à l'exécution nul. Son

inconvénient essentiel est le surcoût de l'espace disque nécessaire à cause de la sauvegarde des différentes versions du document.

Dans le cas dynamique, l'adaptation est réalisée dynamiquement par le serveur, à la demande de l'utilisateur. A l'inverse des caractéristiques de la première méthode (statique), elle aboutit à un surcoût disque nul, mais un coût processeur et une surcharge serveur importants.

En général, cette approche a une flexibilité et une extensibilité limitées ; malgré leur capacité de calcul et de traitement, les serveurs peuvent être saturés suite à un grand nombre de requêtes et risquent de ne plus satisfaire les clients.

2.3.8 Les axes de l'adaptation des Services Web

L'adaptation au contexte dans un Système d'Information Web restituant des Services Web se focalise sur trois axes : l'adaptation du comportement du service, l'adaptation du contenu du service et l'adaptation de la présentation du service, et peut avoir lieu soit avant de démarrer le service, soit durant son exécution. Cette section a pour but de présenter de manière plus détaillée ces trois axes de l'adaptation.

2.3.8.1 Adaptation du comportement du service

Le nombre d'utilisateurs du réseau Internet ne cesse de croitre depuis les années 90. Les utilisateurs du Web deviennent exigeants en termes d'attente et de qualité. C'est pourquoi des travaux essaient d'adapter le comportement des services afin qu'ils préviennent les changements de l'environnement d'exécution. L'adaptation du comportement du service consiste à changer la logique d'un service.

Elle concerne l'occurrence et l'ordre des étapes d'exécution d'un service, ainsi que les relations logiques entre ces différentes étapes [18]

Plusieurs travaux se concentrent sur l'adaptation du comportement du service. Parmi ces travaux, nous pouvons citer :

- Les travaux de *Riveill et al, Kazi-Aoul et al***:** [19]
  - Ces travaux sont basés sur un processus visant à adapter les paramètres d'un service ou les composants du service en fonction des changements de l'environnement de l'utilisateur. Un service est conçu comme un assemblage de composants interconnectés, capables d'être supervisés et reconfigurés si

nécessaire. Un composant est une unité d'encapsulation des fonctions et de déploiement qui offre une ou plusieurs interfaces pour exploiter ses fonctionnalités et pour exprimer ses dépendances fonctionnelles. Ces composants peuvent être personnalisés et assemblés avec d'autres composants à travers des connecteurs sans modification de leur code (Figure 3.2).

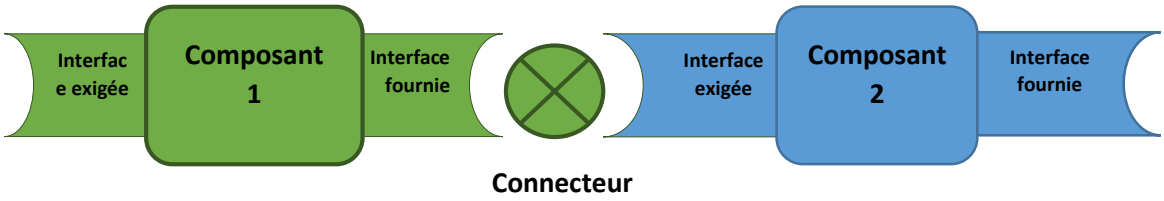

Figure 2.3 : «Assemblage des composants»

- *Les travaux de Brian d m et Marquet et al* [20]
  - Ceux-ci se sont concentrés sur l'adaptation de la ***logique de service*** dans ce cas-là le service est représenté par un ensemble des composants, l'adaptation vise à modifier l'assemblage de ces composants en ajoutant, en retraitant ou en remplaçant un composant.
- *Les travaux de Aksit, Ketfi , et Kazi-Aoul et al*
  - Dans ces travaux l'adaptation du comportement du service peut se faire en modifiant l'assemblage de composants du service par ajout, retrait ou remplacement des composants. L'action peut également se faire sur la distribution des composants.

Les actions possibles sont : [19]

Ajouter un nouveau composant

Cette adaptation implique l'adaptation des interconnexions entre les composants en ajoutant un composant pour que le service puisse s'adapter aux changements du contexte de son exécution. Quand un composant est ajouté à un service en cours d'exécution, il doit donc récupérer l'état à partir duquel il doit démarrer et se personnaliser en fonction de cet état.

Supprimer un composant existant

Cette adaptation vise à supprimer un des composants du service pour que ce dernier soit adapté aux changements du contexte. Le composant supprimé ne doit pas affecter l'exécution des autres composants. Il doit être aussi dans un état stable avant d'être

supprimé. Par exemple, si un composant est en cours d'écriture de données dans un fichier, il ne doit pas être supprimé avant la fin de sa tâche. Un autre défi concerne les données et les messages échangés entre ce composant et les autres, ces données ou messages ne doivent pas être perdus.

- **Remplacement de composants**

Cette adaptation vise à supprimer un composant et en mettre un autre à sa place afin de rendre le service adapté aux changements du contexte. Le nouveau composant doit être capable de se connecter aux composants existants sans affecter ses exécutions. [19]

- **Modifier les interconnexions entre composants**

Par exemple, quand un nouveau composant est ajouté, il doit être connecté aux composants existants. En général, lorsque deux composants sont connectés, les types de leurs ports connectés doivent être compatibles. La communication entre deux composants dans le même espace d'exécution est différente de la communication entre deux composants dans deux espaces d'exécution différents sur la même machine. Elle est également différente de la communication entre deux composants s'exécutant sur deux machines différentes. L'adaptation des interconnexions doit préserver les messages et les données en transit.

- *Les travaux de Cremene et al [19]*
  - Les auteurs ont proposé une architecture logicielle qui rend possible l'adaptation dynamique des services construits par assemblage de composants, en fonction d'un contexte varié.

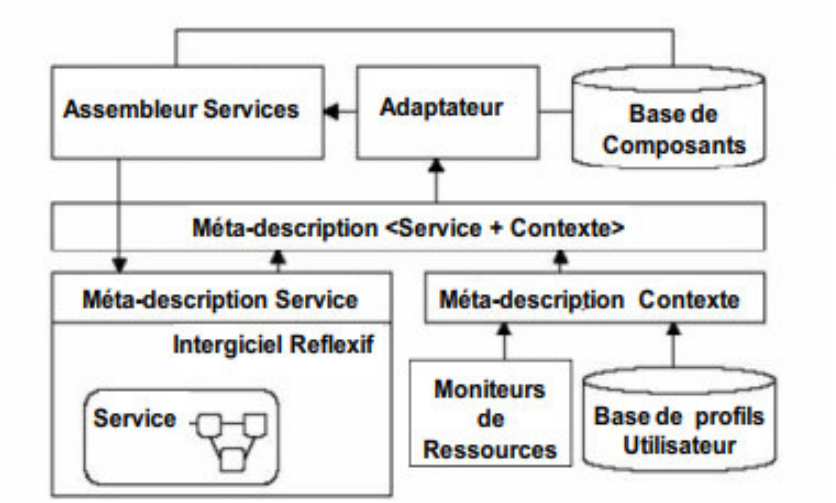

Figure 2.4 : «Architecture pour la composition des services selon le contexte» 19]

Ce contexte concerne les besoins de l'utilisateur. Des profils qui décrivent non seulement les éléments du contexte (méta-description contexte) mais aussi chaque composant constituant le service (méta-description contexte) sont utilisés par l'Adaptateur afin de confirmer la conformité entre les différents profils de chaque composant et les profils des éléments du contexte. L'Adaptateur détecte les points d'inadaptation, cherche et applique aux différents composants du service les modifications nécessaires pour rétablir cette compatibilité par ajout, retrait ou remplacement de composants ,Figure 2.4.

L'adaptateur appelle un composant qui répond au même objectif, modifie le programme de composant afin qu'il réponde aux attentes de l'utilisateur, etc.

- *Les travaux de Cremene et al* [19]
  - Pour ces derniers, chaque service doit être polymorphe et doit donc posséder plusieurs versions ou instanciations possibles suivant la situation contextuelle courante. Nous définissons une entité « adaptateur » （adapter） qui permet d'effectuer le choix de la version ou de l'instance adéquate.

- *Les travaux de Boszomenyi l., et al* [20]
  - Les auteurs se sont concentrés sur l'adaptation du contenu du service ; il s'agit de l'ensemble des formes d'adaptabilité qui choisissent le changement du contenu fourni par le service selon le contexte. Une forme typique de l'adaptation du contenu est le changement de l'encodage d'un flux multimédia ou le changement de la présentation d'un service selon le contexte.
- *Les travaux de Kazi et al* [20]
  - Dans leur travail, les auteurs ont proposé une architecture générique pour la fourniture de services adaptables aux usagers de terminaux mobiles. Cette architecture s'attaque simultanément à l'adaptation de la logique de services en utilisant des composants et à l'adaptation des flux multimédia en s'appuyant sur les travaux de MPEG-7. Une base de flux multimédia et une base de services et/ou des composants de services sont générées par le prestataire de services. Ces deux bases contiennent des versions des données et des composants afin de choisir la version pertinente du service au contexte de l'utilisateur.

- *Les travaux de Fouial et al* [18]
  - L'adaptation du comportement du service consiste à changer la logique d'un service. Elle concerne l'occurrence et l'ordre des étapes d'exécution d'un service, ainsi que les relations logiques entre ces différentes étapes.
  - Ont présenté une architecture qui vise à adapter dynamiquement des services au contexte de l'utilisateur en considérant que chaque service est constitué d'un ensemble de composants interconnectés. L'adaptation des services est réalisée au moment de l'accès au service et au moment de son exécution sur le terminal. [19]
- L'adaptation au moment de l'accès au service consiste à sélectionner et assembler les composants du service en fonction du contexte. Pour réaliser l'adaptation du service au moment de son exécution, plusieurs versions des composants du chaque service sont sauvegardées. Pour chaque variation dans l'environnement de l'utilisation du service, un changement dynamique de la composition du service est réalisé en modifiant les versions du composant, en retirant ou en remplaçant des composants du service. [16]

2.3.8.2 Adaptation du contenu du Service

L'adaptation du contenu consiste à modifier les propriétés des données présentées à l'utilisateur intéressé. Elle peut être illustrée par la modification du type des données aux capacités du terminal, aux capacités du réseau et/ou aux préférences de l'utilisateur. Par exemple, la transformation du contenu peut consister à transformer un texte en une synthèse vocale. [20]

Ce contenu est constitué de données de type multimédia (texte, audio, image, vidéo). Les modifications nécessaires afin de réaliser l'adaptation sont assurées par un ensemble de services d'adaptation des données multimédia. [19]

Nous distinguons trois techniques d'adaptation :

- **Le transmodage**

  Appelé également conversion de modalités, réfère au changement de mode (ou de modalité) d'un média .il s'agit, par exemple, de transformer un texte en image de ce texte pour un client qui ne dispose pas de la police de caractère permettent l'affichage de ce texte.

- **Le transcodage**

  Consiste à changer le format d'un média donné sans changer de modalité, de telle sorte que le contenu d'origine garde le même aspect. Un exemple serait celui de convertir une vidéo de format MOV au format vidéo AVI.

- **La transformation**

  Se réfère à tout changement d'un média donné sans modifier ni sa modalité ni son format, Ce processus transforme un contenu en réduisent sa taille par exemple, ou en traduisant du texte. Cette catégorie concerne également des adaptateurs spécifiques qui reconstruisent un document multimédia de synchronisation de type SMIL ou MPEG -21 par exemple et qui prennent en entrée un document multimédia et le transforment en un autre document multimédia.

	Transmodage	**Transcodage**	**Transformation**
Vidéo	➢ Vidéo vers image ➢ Vidéo vers slideshow ➢ Vidéo vers texte	➢ Convertisseur de format vidéo	➢ Redimensionnement de vidéo ➢ Changement de couleur ➢ Restauration de couleur ➢ Réduction de bruit ➢ Compression de vidéo ➢ Résumé ➢ Changement de résolution
Image	➢ Image vers vidéo ➢ Texte vers vidéo	➢ Convertisseur de format image	➢ Compression ➢ Changement de résolution Spatiale ➢ Changement de couleur
Audio	➢ Audio vers texte	➢ Convertisseur de format audio	➢ Compression d'un contenu audio ➢ Résumé
Texte	➢ Texte vers slideshow ➢ Texte vers audio	➢ Convertisseur de format texte	➢ Traduction ➢ Résumé ➢ Suppression d'espaces ➢ Changement de la taille du texte

Tableau 2.1 : «Classification de quelques techniques d'adaptation» [19]

Ainsi, le processus d'adaptation du contenu s'applique sur une forme du contenu et produit en sortie une nouvelle forme alternative. Ce processus peut être appliqué à différents niveaux.

Nous distinguons trois niveaux possibles selon le chemin de transmission du contenu du serveur vers le client cible : niveau serveur, niveau client et niveau intermédiaire. Par conséquent le contenu adapté peut être le résultat de plusieurs tâches d'adaptation appliquées à plusieurs niveaux. [16]

En fait, il n'y a pas beaucoup de travaux réalisés liés à l'adaptation du contenu du Service Web par rapport au contexte. Parmi les travaux effectués, nous pouvons citer les travaux de :

Les travaux de Brusilovsky [18]

- Il existe trois manières courantes d'adapter le contenu :
  - Fournir un supplément d'information.
  - Cacher des données jugées non pertinentes ou ne devant pas être délivrées.
  - Choisir le contenu adéquat à l'utilisateur, si le système possède plusieurs alternatives de contenu.

Les travaux de Lemlouma [19]

- L'adaptation du contenu est définie généralement comme le processus qui transforme un contenu de son état initial vers un état final afin de satisfaire un ensemble de contraintes (des caractéristiques, des buts de l'utilisateur, etc.). Le contenu final peut provenir du même contenu source ou d'autres ressources de contenu

Les travaux de Pashtan et al [16]

- Dans ces travaux, le contenu des résultats des Services Web est adapté aux dispositifs mobiles et à l'utilisateur.

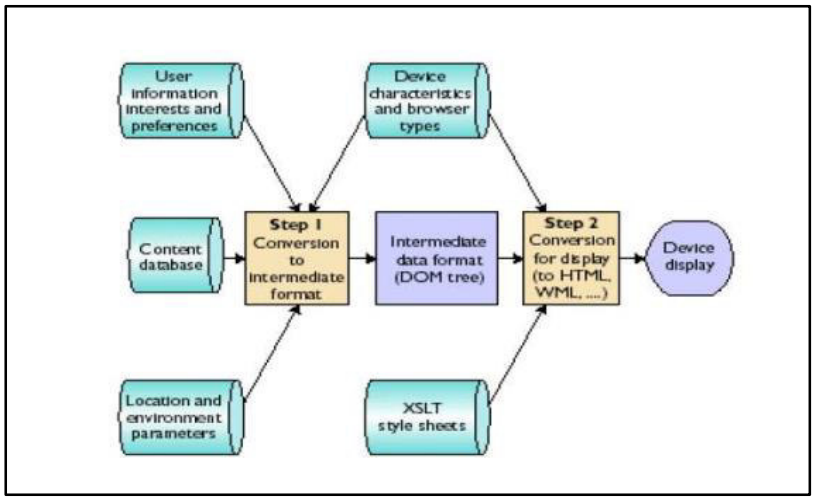

Figure 2.5 : « Conversion dynamique de contenu du résultat d'un Service Web »

Afin d'adapter le contenu du résultat du service appelé, un processus de génération du contenu de ce service en un format intermédiaire est utilisé (Figure 2.5).

Le résultat du Service Web appelé est sauvegardé dans une base de données (Content Data base). Ce résultat est converti dans un format intermédiaire selon le contexte. Ce dernier est stocké dans trois bases de données :

1) Une base de données contient les informations et les préférences de l'utilisateur,

2) Une deuxième inclut les données concernant la localisation de l'utilisateur et son environnement,

3) Une troisième contient des caractéristiques des dispositifs et du navigateur.

Les travaux de Fouial [16]

- A proposé une application MADSSERV (Media ADaptive Streaming Service). Cette application vise à adapter un contenu multimédia en particulier vidéo au contexte concernant le terminal mobile. Cette adaptation consiste à modifier la qualité de vidéo afin d'assurer une fourniture de services sensibles au contexte.

L'application MADSSERV effectue l'adaptation sur des nœuds intermédiaires, placés entre les fournisseurs qui proposent leurs services et les terminaux mobiles qui exécutent ces services. Sur ces nœuds, des mécanismes d'adaptation sont appliqués aux données multimédia et permettent de prendre en considération les variations dans le contexte.

Le gestionnaire d'adaptabilité impose le format final de contenu multimédia. Pour cela, il prend en paramètres d'entrée le niveau de qualité (transmis par l'utilisateur de choix le niveau de qualité) ainsi que le format original du service multimédia. A partir de ces informations, le gestionnaire d'adaptation détermine le niveau de qualité optimale.

Le gestionnaire d'adaptabilité travail selon plusieurs niveaux de qualité : [19]

- **Niveau 1** : visualiser la séquence vidéo en noir et blanc ;
- **Niveau 2** : n'envoyer que le son avec des images extraites de la vidéo à des intervalles de temps régulière ;
- **Niveau 3** : n'envoyer que le son ;
- **Niveau 4** : envoyer du texte si la transcription de la piste sonore est disponible.

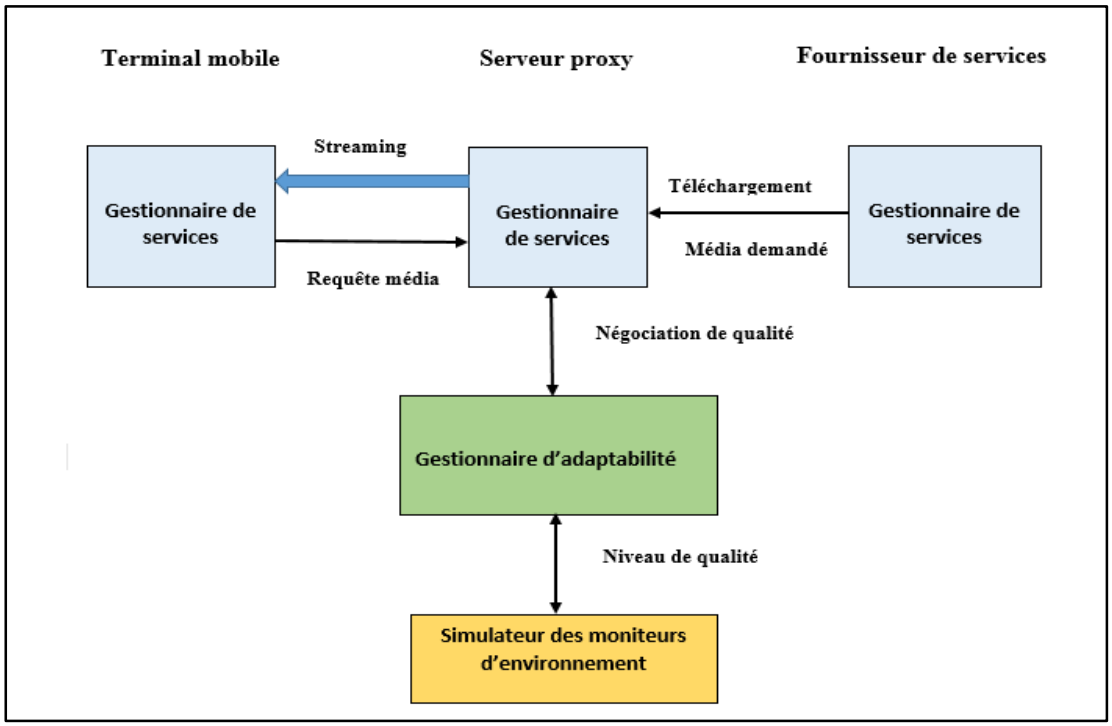

Figure 2.6 : « Module développés pour assurer l'adaptation des services » [19]

2.3.8.3 Adaptation de la présentation du service

L'adaptation de la présentation est appliquée sur les caractéristiques visuelles du document présentées à l'utilisateur. Elle vise à effectuer des transformations au niveau de l'affichage.

Ces transformations suivent les préférences de l'utilisateur et son contexte. Nous décrivons les méthodes les plus courantes pour effectuer cette adaptation :

- L'adaptation des caractéristiques graphiques ;
- L'adaptation de la langue de présentation ;
- L'organisation du document ;
- L'altération de médias et la substitution de médias.

Nous décrivons les méthodes les plus courantes pour effectuer cette adaptation :

❖ *Koch et Frasincar et al* [19]

➢ L'adaptation des caractéristiques graphiques consiste à intervenir sur les choix de couleurs, de formes, de taille, de composant, etc.

➢ *Koch et Frasincar et al*

L'adaptation de la langue de présentation consiste à choisir le langage utilisé pour diffuser l'information

❖ *Frasincar et al* [16]

➢ L'organisation du document concerne l'organisation spatiale ou temporelle du document.

Dans cette méthode, la page est décomposée en différentes régions qui peuvent être positionnées différemment dans la page et activées selon des ordonnancements temporels variés.

❖ *Martin et al*

➢ L'altération de médias et la substitution de médias sont des méthodes concernant les données multimédias. La première vise à des adaptations qui modifient une information délivrée par un média sans changer de support : la réduction de la taille d'une image, le passage d'une image couleur à une image noir et blanc, le remplacement d'une vidéo par un résumé vidéo, etc. La deuxième méthode permet de remplacer une information, supportée par un média donné, par une autre représentation de cette information en utilisant un autre type de média. Par exemple, la substitution à une vidéo d'une image extraite de celle-ci.

Plusieurs techniques existent pour mettre en œuvre de telles adaptations de présentation. Technique basée sur l'utilisation des feuilles de styles. En effet, la création de différentes feuilles de style – qui incluent les spécifications des adaptations désirées – permet de produire différents documents bien que les feuilles de style soient appliquées à un même contenu de départ. L'intérêt est ici de pouvoir définir une feuille de style pouvant s'appliquer à plusieurs documents. [19]

- *Pashtan et al*
  - Par exemple ont utilisé les feuilles de style XSL (Extensible Stylesheet Language) ou XSLT (XSL Transformations) afin de prédéfinir et adapter si besoin la présentation du résultat du service.

Keidl et al [19]

- Ont proposé d'inclure dans le modèle de l'utilisateur un ensemble de feuilles de style en adéquation avec le terminal de l'utilisateur.

  En ce qui concerne l'implantation des méthodes de données de type multimédia, [16] des programmes spécifiques de traitement d'images, de construction de résumés de vidéos, etc. Sont employés. Des techniques moins compliquées peuvent cependant être utilisées, comme la sélection d'une version d'information (i.e. version vidéo d'une information, sa représentation en images clés ou sous la forme d'une description textuelle du contenu) parmi un ensemble créé et stocké au préalable.

- *Pashtan et al*
  - Par exemple adaptent la présentation des résultats des Services Web à l'utilisateur et à son dispositif mobile. Afin de réaliser cette adaptation, un parseur XML DOM est utilisé pour convertir le contenu du service en format HTML ou en format WML adapté aux caractéristiques du dispositif de l'utilisateur et au navigateur utilisé. Ce processus de conversion pour réaliser l'adaptation de la présentation est établi par des feuilles de style XSLT (XSL Transformations).

Discussion

Nous avons présenté les différentes formes d'adaptation dynamique du service aux changements de l'environnement de l'utilisateur : l'adaptation du comportement du service, l'adaptation du contenu du service et l'adaptation de la présentation du service.

Dans la première forme d'adaptation, nous remarquons que les travaux proposés sont basés sur la sauvegarde de plusieurs versions des composants du service en choisissant la version compatible avec les variations de l'environnement d'utilisation du service ou sur la sauvegarde des composants afin de remplacer le composant par un autre

composant plus compatible avec le contexte. L'inconvénient est que l'utilisation de ces méthodes a abouti à la surcharge du système.

La majorité des travaux présentés dans la section concernant la deuxième forme d'adaptation est basée sur l'utilisation d'une approche de sélection de versions, qui est basée sur la sauvegarde de plusieurs versions du contenu du service afin de choisir la version compatible avec les variations de l'environnement d'utilisation du service, conduit aussi à une surcharge du serveur. Nous remarquons aussi que la plupart des travaux présentés dans une approche complexe :

- N'amènent pas de solutions générales à tous les types de média utilisés mais essaient de fournir des solutions à des besoins très spécifiques tels que l'adaptation des images pour les mobiles, le transcodage de la vidéo, etc. Par exemple les travaux de [Fouial, 2004] portent sur l'adaptation au contexte d'un contenu multimédia en particulier vidéo demandé par un terminal mobile. Dans les travaux de [Pashtan et al., 2004], l'adaptation du contenu est basée sur une conversion du résultat où ce résultat est toujours présenté en mode texte. Les données multimédia ne sont pas prises en considération.
- N'utilisent pas toutes les informations contextuelles pour adapter le service : ces services ne sont adaptés qu'aux ressources réseaux et/ou aux capacités du terminal. Par exemple *Fouial*, essaie d'adapter un type de média pour l'utilisation des dispositifs mobiles ayant des capacités limitées.

Le tableau 3.2 synthétise les aspects principaux (type d'adaptation, contexte, méthode d'adaptation) des systèmes étudiés concernant les trois types d'adaptation.

Système	**type d'adaptation**	**Contexte par rapport**			**Méthode d'adaptation**
		À l'utilisateur	Au dispositif Autres	Au service	
[Cremene et al., 2004]	Comportement	Besoins de l'utilisateur	/	Utilise le contexte de service	Ajout, retrait ou remplacement de composants
[Fouial et al., 2002]	Comportement	/	Caractéristiques du terminal	Les caractéristiques d'un service	Sauvegarde de plusieurs versions des composants du service
[Pashtan et al., 2004]	Contenu et présentation (texte)	Préférences	Caractéristiques du dispositif mobile de l'utilisateur	/	Convertir les résultats du service selon le contexte
[Fouial et al., 2004]	Contenu (vidéo)	/	Caractéristiques du dispositif mobile de l'utilisateur	/	Approche visant à améliorer et/ou à dégrader de la qualité du service
[Tarak Chaari]	Comportement	Besoins les caractéristiques de terminale	/	/	Sauvegarde plusieurs versions d'un service web et l'adaptateur effectue le choix d'une version compatible avec le contexte d'utilisateur

Tableau 2.2 : **« Comparatifs entre les approches d'adaptation »** [16]

2.10 Avantages et inconvénients

Ce tableau a pour objectif de présenter les avantages et inconvénients de chaque approche d'adaptation : [

L'approche d'adaptation	Avantages	Inconvénients
➢ **Comportement**	➢ Produire un nouveau service compatible avec la machine d'utilisateur.	➢ Base des composants 1. La surcharge des systèmes. 2. La possibilité d'adaptation dans ce domaine à des remplace, ajout ou supprime d'un composant.
➢ **Contenu**	➢ Adapte le résultat d'un service web (image, vidéo, texte).	➢ Les types d'adaptation de contenu qui sont focalisés sur la transformation d'image ou le transcodage vidéo, et ne fournissent pas de solution générique d'adaptation.
➢ **Présentation**	➢ Produire des interfaces d'interaction entre l'utilisateur et le service.	➢ Approches existantes restent limitées à des plateformes spécifique.

Tableau 2.3 : **«Avantages et inconvénients»** [19]

2.4 Service Web sensible au contexte d'utilisation

Certains travaux ont traité la problématique de fournir à l'utilisateur un service adaptable selon différents points de vue : préférences, terminal, réseau,…..etc.

- *Bellavista et al* [20]
  - A proposé des plateformes qui s'intéressent à l'applicabilité des agents mobiles pour la personnalisation des services et la nomadicité des utilisateurs dans les environnements mobiles. Le contexte est caractérisé par les capacités du terminal.
- *Vesper et al*
  - Réalisent la modélisation et l'exploitation d'applications de service à utilisateur potentiellement mobiles. Le contexte est souvent représenté soit par les ressources réseau, soit par les capacités du terminal.
- *Cheverest et al et Cameleon et al*
  - A proposé un service qui délivre à l'utilisateur des informations pertinentes à son contexte. Dans ces travaux, le contexte est présenté par la localisation de l'utilisateur et ses préférences.
- *Kassab et al* [17]
  - L'information sur les préférences peut être reconnue par un système selon les trois

    manières suivantes :
    - Elle peut être fournie par l'utilisateur.
    - Les concepteurs des systèmes d'information peuvent l'avoir définie à travers de profils généraux d'utilisateurs.
    - Le système peut la déduire d'historique de l'utilisateur.

- *Carrillo et al*[17]
  - Proposent PUMAS (Peer Ubiquitous Multi-Agent System), un Framework qui récupère l'information distribuée entre plusieurs systèmes d'information web et/ou accédée à travers différents types de Dispositifs Mobiles. Ce qui nous a attiré dans ce travail est le concept de « préférences utilisateur » : un système de gestion de préférences consiste en la capture du contexte d'utilisation de la session et en la sélection des préférences (d'activité, de Résultat et d'affichage) qui peuvent être appliquées dans les circonstances de la session en cours.

- les préférences sont classées en trois catégories :
  - **Préférences d'activité** : ce sont les tâches et les fonctionnalités (activités) qu'un utilisateur envisage d'accomplir dans le système et la manière de le faire (séquentielle, concurrente, conditionnelle).
  - **Préférences de Résultat** : spécifient le type et l'ordre des résultats des activités.
  - **Préférences d'affichage** : désignent le mode de représentation et la manière dont l'utilisateur souhaiterait que l'information soit affichée.

❖ *Chaari* [17]

- Propose une stratégie d'adaptation d'applications au contexte d'utilisation sur trois volets : **(i)** les services offerts à l'utilisateur, **(ii)** les données renvoyées par ces services et **(iii)** leur présentation à l'utilisateur. Il a mis en œuvre cette stratégie en développant une plateforme d'adaptation d'applications au contexte d'utilisation dite SECAS (Simple Environment for Context-Aware Systems).

2.11 Conclusion

De nos jours, la diversité des dispositifs employés par l'utilisateur pour accéder aux ressources (PC, ordinateurs portables, PDA, etc.) en utilisant plusieurs types des réseaux (Wifi, local, etc.) engendre un besoin croissant d'adapter dynamiquement des services au contexte de l'utilisateur. Donc, afin d'assurer l'utilisabilité du service, il est nécessaire de progresser dans la phase de conception de ce service en trouvant des moyens pour réaliser l'adaptation au contexte de l'utilisateur. Ce contexte doit donc contenir des informations sur l'environnement de l'utilisateur (le dispositif, le réseau, la localisation, etc.) et son profil (les préférences, nom, Etc.).

Dans notre travail, nous nous focalisons sur l'adaptation côté client afin d'adapter le résultat d'un Service Web au contexte en utilisant une méthode dynamique (pour plus des détails, vous pouvez vous référer chapitre 5).

3.1 Introduction

La technologie des agents fait partie du domaine de l'intelligence artificielle distribuée. dans celle-ci, l'intelligence et l'expertise sont distribuées sur un ensemble d'entités logicielles ou physiques autonomes, en interactions, qui travaillent en collaboration pour résoudre un problème ou attendre certains objectifs. Cette technologie a tiré profit d'autres disciplines comme la sociologie, la psychologie sociale et les sciences cognitives.

Aujourd'hui la technologie basée agents est largement utilisée et les agents sont impliqués dans plusieurs applications informatiques. Plus particulièrement, celles caractérisées par la recherche d'atteindre certains buts, puisqu'un un agent est une entité qui agit à la demande de quelqu'un pour accomplir quelque chose.

3.2 Intelligence artificielle distribuée

L'expression Intelligence Artificielle (IA) est employée pour la première fois (1955-1970) par John McCarthy. TI fonde l'Intelligence Artificielle sur le postulat mécaniste qui veut que toute activité intelligente soit modélisable et reproductible par une machine.[21]

« L'IA a pour but de faire exécuter par l'ordinateur des tâches pour lesquelles l'homme, dans un contexte donné, est aujourd'hui meilleur que la machine » [ALL94].

L'Intelligence Artificielle Distribuée (IAD) est un sous-domaine d'Intelligence Artificielle qui s'occupe des situations où plusieurs systèmes interagissent pour résoudre un problème commun [MOU96b]. L'IAD se divise en deux branches principales :

- La Résolution Distribuée de Problèmes (RDP) qui étudie comment distribué des compétences au niveau de chaque partie du système, de façon à ce qu'il soit globalement plus compétent que chacune de ses parties ;
- La Simulation des Systèmes Complexes (SSC), qui concerne plus particulièrement les Systèmes Multi-Agents (SMA). Les SMA traitent le comportement d'un ensemble d'agents autonomes qui essaient de résoudre un problème commun.

La différence notable entre la RDP et les SMA est que la RDP possède une approche descendante (« top-down») et les SMA une approche ascendante («bottom-up»)

3.3 Développement des réseaux et systèmes distribués

La généralisation des réseaux, la coopération entre plusieurs composants logiciels au sein d'environnements hétérogènes et distribués et le développement d'internet ont ouvert la voie à de nouvelles applications SMA. Dans le cas de réseaux, on trouve les travaux auteur de la gestion de réseaux, les réseaux intelligents, etc. [22]

Dans le domaine d'internet, les agents intelligents sont de plus en plus utilisés pour offrir de meilleurs services aux utilisateurs (par exemple recherche intelligente, assistance, commerce électronique, etc.). On peut citer également toutes les applications liées à la coopération des logiciels, des composants distribués, etc.

Plusieurs travaux dans le domaine des systèmes distribués introduisent des concepts d'IAD tels que la répartition du contrôle, le concept de connaissance distribuée (telle que définie dans, etc. pour la conception d'applications réparties et coopératives, Cet intérêt pour les SMA est dû essentiellement au fait qu'une telle approche permet :

- La décomposition et la répartition des connaissances et des mécanismes de traitement dont l'unité de base est l'agent.
- Le caractère dynamique du contrôle de résolution d'un problème distribué caractérisée par l'organisation dynamique et l'affectation des tâches modifiables en cours de résolution.
- L'aptitude à traiter des problèmes simultanés et potentiellement corrélés avec des optimisations éventuelles.
- L'adaptabilité et la possibilité d'apprentissage des agents qui leur confèrent la capacité de résister à des environnements évolutifs et/ou instables.

3.4 Notion d'agent

Bien qu'il y ait peu de consensus autour des concepts agents, plusieurs chercheurs ont des définitions qui convergent vers celles-ci. [21]

3.4.1 Définition

Le concept d'agent comme plusieurs autres concepts est défini de plusieurs manières différentes, quoi que toutes les définitions de l'agent ont quelques points en commun. La communauté scientifique a accepté des définitions parmi lesquelles nous citons la définition de Ferber considérée l'une des premières définitions où *« Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son*

environnement, qui, dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents » .

Jennings propose la définition suivante : *« Un agent est un système informatique qui est situé dans un certain environnement et qui est capable d'effectuer de manière autonome une action afin de répondre aux objectifs pour lesquels il a été conçu ».*

En synthétisant toutes les définitions proposées pour le concept d'agent, il en résulte qu'un agent est une entité physique ou virtuelle ayant les qualités suivantes: l'agent est capable d'agir dans un environnement ; il peut communiquer directement avec d'autres agents ; son comportement est contraint par des objectifs individuels ; Il possède des ressources propres ; Il est capable de percevoir son environnement et dispose d'une représentation partielle de cet environnement ; l'agent possède des compétences et offre des services ; Il a la capacité de se reproduire ; le comportement d'un agent tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont il dispose, et en fonction de sa perception, de ses représentations de l'environnement ainsi que les communications qu'il effectue avec les autres entités.

La figure 3.1 donne, de façon générale, l'architecture interne d'un agent.

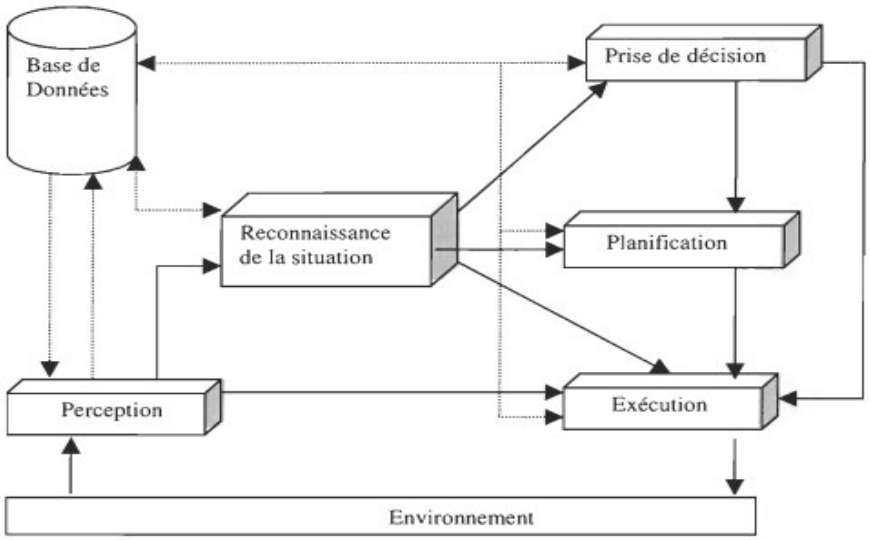

Figure 3.1 : « Architecture interne d'un Agent »

3.4.2 Caractéristiques d'un agent

Les caractéristiques principales qu'un agent possède et qui sont pratiquement objet de consensus par toute la communauté agent sont définis par Jennings [24] : Agents & Composants : Concepts, analyse comparative et apports mutuels.

- **Situation** : L'agent est une entité située, capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement ;
- **Autonomie** : L'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne ;
- **Apprentissage** : Un agent est capable d'apprendre et d'évoluer en fonction de cet apprentissage, il est aussi capable de changer le comportement en fonction des expériences passées ;
- **Mobilité** : La capacité d'un agent de se déplacer à travers un réseau d'une machine à une autre ;
- **Interactivité :** L'agent doit pouvoir des actions sur son environnement et réciproquement ;
- **Délégation :** L'agent accomplit un ensemble de taches a la demande d'un utilisateur ou d'un autre agent ;
- **Compétition :** L'agent est capable d'agir dans un environnement où d'autres agents interviennent .le but est le même pour tous les agents présents mais un seul l'atteindra, les autres échoueront ;
- **Coordination :** L'agent est capable de coordonner ses actions par rapport à un utilisateur
- ou un autre agent ;
- **Flexibilité** : Lette caractéristique résume les propriétés suivantes : [24]
- **Réactivité** : Un agent est capable de modifier son comportement lorsque les conditions environnementales changent et de percevoir son environnement et d'élaborer une réponse dans les temps requis ;
- **Proactivité** : Les agents n'agissent pas seulement en réponse à leur environnement mais ils sont également capables d'avoir un comportement guidé par un but en ayant la possibilité de prendre l'initiative ;
- **Sociabilité** : L'agent doit être capable d'interagir avec les autres agents (logiciels ou humains) et peut se trouver engagé dans des transactions sociales ;

- **Activité** : Un agent est toujours actif ; il s'exécute dans un thread ou un processus indépendant ;
- **Communication :** C'est la possibilité d'échange de messages entre agents, selon l'un des schémas de communication : **Now-type messaging** : dans lequel l'émetteur du message bloque son exécution jusqu'à ce que le récepteur ait téléchargé le message et envoyé sa réponse ; **Future-type messaging** où l'émetteur ne bloque pas son exécution mais l'expéditeur retient une variable qui peut être utilisée pour obtenir le résultat et finalement le schéma **One-way** messaging qui représente un type de messagerie asynchrone qui ne bloque pas l'exécution courante. L'expéditeur ne retient pas une variable pour ce message et le récepteur ne va jamais répondre. Ce type est utile quand deux agents engagent une conversation où l'agent expéditeur n'a pas besoin de la réponse de l'agent récepteur.

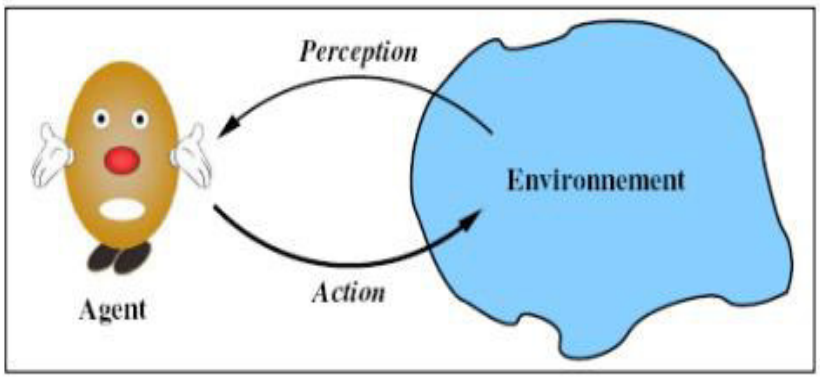

Figure 3.2 : « Un Agent avec son environnement »

3.4.3 Structure d'un agent

Ferber propose une structure générale qu'un agent possède tel que : le savoir-faire, les croyances, le contrôle, l'expertise et la communication. [24]

- **Le savoir-faire** : C'est une interface permettant la déclaration des connaissances et des compétences de l'agent. Il permet aussi, la sélection des agents à solliciter pour une tâche donnée ;
- **Les croyances** : Ce que l'agent connaît du monde (la représentation de son environnement, les autres agents et lui –même). On préfère parler de croyances au lieu de connaissances car dans un système multi-agents les connaissances que l'agent a sur lui-même et sur les autres agents ne sont pas nécessairement

objectives. Cependant, la formalisation de ses connaissances sera à la base de la conception de tout système multi-agents car elle détermine en grande partie le comportement « intelligent » des agents ;

- **Le contrôle** : La connaissance du contrôle dans un agent est représentée par les buts, les intentions, les plans et les tâches qu'il possède ;
- **L'expertise** : C'est la connaissance sur la résolution du problème. C'est la base de règles par exemple pour un système expert ;
- **La communication :** Pour que deux personnes puissent communiquer, ils doivent parler la même langue. Dans le cas des agents, le même principe s'impose c'est pourquoi la création d'un langage commun à tous les agents s'imposait pour assurer une bonne communication et une bonne coordination d'actions. Ce langage n'est autre que l'ensemble de primitives connues par chaque entité dont l'ensemble constituée un contrôle de communication

3.4.4 Approche orientée-agent vs approche orientée-objet

Les deux grandes branches de l'informatique « moderne » sont sans doute la programmation orientée objets (POO) et la programmation orientée agents (POA). [21]

Voici les principales différences entre les deux approches.

- Un agent est un objet qui est adaptatif, rationnel, autonome, capable de communiquer et d'agir.
- Un agent logiciel est de préférence créé sous forme d'un objet ayant les caractéristiques d'un processus.
- Un objet O1 peut appeler une méthode existante d'un objet O2 qui doit l'exécuter.
- Un agent A1 peut demander l'exécution d'une méthode à l'agent A2 qui est libre de l'exécuter ou pas.
- Un agent agit en fonction de son but, de ses contraintes et de ses capacités. Ces points font la différence entre agents et objets, et c'est sur ces bases que se fait le choix de l'approche à utiliser selon la problématique.
- Les objets sont généralement passifs alors que les agents sont permanents actifs ;
- Les agents sont autonomes et responsables de leurs actions alors que les objets n'en sont toujours pas ;
- On ne peut prévoir tous les comportements des agents dans les systèmes ;

- L'approche orientée-objet ne fournit pas un ensemble adéquat de concepts et de mécanismes pour modéliser les systèmes complexes dans lesquels les rapports évoluent dynamiquement ;
- Certains chercheurs définissent un agent comme un objet actif ayant une autonomie et un objectif.

3.4.5 Type d'agents

En fonction de la taille d'un agent, de sa complexité, de ses connaissances et de son raisonnement, nous pouvons classer les approches multi agents en trois grandes catégories : cognitive, réactive et hybride. [22]

3.4.5.1 Agents cognitifs

Il est par lui-même c'est-à-dire qu'il effectue un certain raisonnement pour choisir ses actions. Un tel raisonnement peut se faire soit en se basant sur les buts de l'agent, soit sur une certaine fonction d'utilité.

3.4.5.2 Agents réactifs

Comme son nom l'indique, un agent réactif ne fait que réagir aux changements qui surviennent dans l'environnement. Autrement dit, un tel agent se contente simplement d'acquérir des perceptions et de réagir à celles-ci en appliquant certaines règles prédéfinies. Étant donné qu'il n'y a pratiquement pas de raisonnement, ces agents peuvent agir et réagir très rapidement.

3.4.5.3 Agents hybrides

Chaque agent hybride est caractérisé par la notion de couches et chaque couche représente soit les agents cognitifs, soit les agents réactifs. Donc l'agent hybride combine entre les deux comportements (comportement réactif et comportement cognitif)

3.4.6 Mobilité

3.4.6.1 Agents fixes

C'est le système le plus immédiat (facile) à implémenter. Un SMA avec des agents non mobile présente tous les intérêts d'un SMA classique : [25]

- Exécution des divers agents en parallèle (en général sous forme de threads) et donc indépendance d'exécution des différents agents.

- Communication grâce à un protocole et un système de communication (en réseau si c'est le cas). Libre arbitre des agents : ils décident de répondre ou non aux sollicitations (messages) extérieures.
- Les agents peuvent êtres de n'importe quel type : réactif, cognitif, etc.

3.4.6.2 Agents mobiles

Lors de l'utilisation d'agents mobiles toutes les caractéristiques des agents fixes sont conservées. L'utilisation des agents mobiles présente en outre plusieurs avantages :

- **De la charge de calcul :**
  - Un agent mobile peut en effet se déplacer sur un ordinateur plus puissant pour effectuer un calcul complexe.
  - Il peut quitter une machine qui est sature pour aller sur une autre.
- **Réduction du trafic réseau :**
  - Un agent qui a besoin de traiter une grande quantité de données situées sur un autre ordinateur (base de données par exemple) peut se déplacer sur l'ordinateur possédant les données et revenir avec le résultat. - Cela permet d'éviter de faire transiter les données sur les réseaux.

Les agents mobiles présentent des intérêts évidents, mais sont cependant plus complexes à gérer. Par exemple :

- Il faut faire suivre les messages lorsque l'agent se déplace.
- Pour qu'un agent puisse se déplacer, il faut bien entendu que des sites d'accueil existent.
- L'utilisation de plusieurs langages de programmation peut poser des problèmes pour la mobilité des agents.

3.4.7 Le rôle des agents :

- Le rôle est une représentation abstraite d'une fonction ou d'un service propose par un agent.
- Un rôle peut être attribue dynamiquement a un agent.
- Chaque méthodologie peut appréhender le rôle de différentes façons. Certaines proposent d'associer potentiellement plusieurs rôles à un agent. D'autres spécifient qu'un rôle est au contraire tenu par plusieurs agents. [25]

3.4.8 Le comportement :

- Un comportement est une réponse à un évènement ou une situation.
- Un évènement est une chose qui se produit et qui change l'environnement ou l'état de l'agent.
- Un agent définit son comportement en fonction des événements qui lui arrivent.
- Lorsqu'un évènement se produit, l'agent doit l'analyser et l'évaluer pour produire une réponse adaptée.
- La décision d'un agent va donner lieu à une action, cette dernière n'est pas seulement envisagée comme le résultat de ce que font les agents mais comme le résultat des réactions du monde aux influences des agents.
- Le comportement interne d'un agent exprime quand et comment un agent va utiliser ses connaissances, ses savoir-faire et ses facultés de perception de l'environnement, ou de communication pour décider de ses actions. Pour un concepteur, la définition du comportement est alors : "comment assembler les différentes parties d'un agent de manière qu'il accomplisse les actions que l'on attend de lui ? ".[25]

3.5 Système multi-agents

3.5.1 Définition

Les systèmes multi-agents font partis de l'intelligence distribuée (IAD). L'idée de base est de distribuer l'intelligence sur plusieurs agents où chaque agent est associé à un sous problème ou sous-objectif. De la coordination des activités de ces agents émerge le but à atteindre ou objectif initial (résoudre le problème). [26]

Ceci a conduit à passer de descriptions de comportements individuels à des comportements collectifs à travers la coopération de plusieurs agents qui sont capables d'associer leurs efforts pour accroître leur intelligence collective. L'IAD s'intéresse donc aux comportements intelligents, résultant de l'activité coopérative de plusieurs entités (agents) .

Le concept de coopération est indispensable pour l'IAD tout comme l'interaction entre les agents qui se traduit par la communication et la négociation.

Donc l'IAD introduit le concept du système multi agents qui est un système constitué d'agents, qui interagissent dans et au travers d'un environnement selon certaines relations, et dont les caractéristiques sont : La coopération, la coordination et la communication.

Ferber a donné plus de détails sur la composition du SMA en précisant qu'un système multi-agents doit être composé des éléments suivants :

- Un environnement E, c'est-à-dire un espace disposant généralement d'une métrique.
- Un ensemble d'objets O. Ces objets sont situés, c'est-à-dire que, pour tout objet, il est possible, à un moment donné, d'associer une position dans E. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.
- Un ensemble A d'agents, qui sont des objets particuliers (A est contenu dans O), lesquels représentent les entités actives du système.
- Un ensemble R de relations qui unissent des objets (et donc des agents) entre eux.
- Un ensemble d'opérateurs Op. Permettant aux agents de l'ensemble A de percevoir, produire, consommer, transformer et manipuler des objets de l'ensemble O.
- Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on pourrait appeler « les lois de l'univers ».

☞ **Exemple**

Les fourmis sont un exemple assez courant de système multi-agents. Ces charmants insectes ont la capacité de trouver le chemin le plus court entre deux points sans avoir a priori de notion de distance. Les données du problème de base dans le cas des fourmis sont que le chemin cherché doit relier la fourmilière à une source de nourriture, deux chemins de longueurs différentes étant envisageables, mais les fourmis n'ayant aucune mesure ni notion de distance. Les fourmis partant de la fourmilière vont prendre indifféremment l'un ou l'autre chemin, à l'aller comme au retour. Sauf que, au retour, ces dernières déposeront sur le chemin choisi une trace chimique (des phéromones), qui aura pour effet d'inciter les autres fourmis à suivre la piste marquée. Et les dépôts successifs auront pour effet de renforcer ce marquage. [27]

Or, le chemin le plus court étant aussi le plus rapide, donc il ne sera vite fait marqué que l'autre voie. Les fourmis étant d'autant plus attirées par un chemin fortement marqué de phéromones, et le phénomène va en s'amplifient.

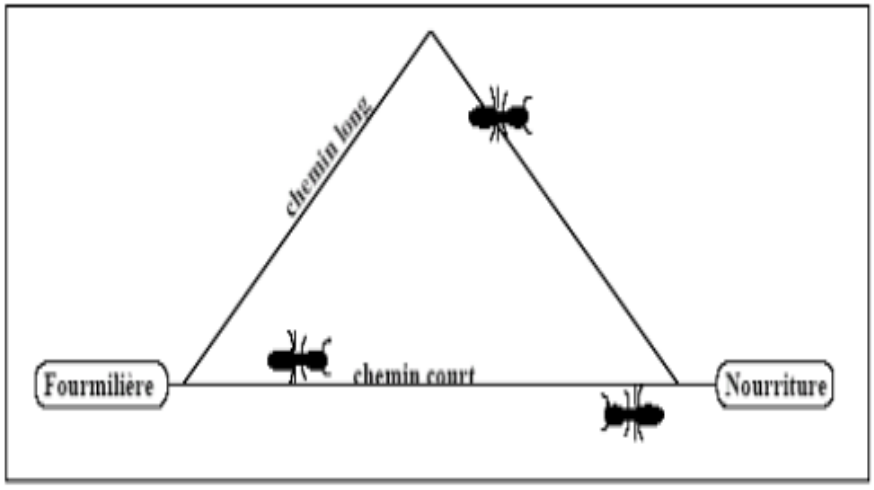

Figure 3.3 : « Recherche du chemin le plus court chez les fourmis » [27]

3.5.2 Pourquoi s'intéresser aux systèmes multi-agents ?

Les systèmes multi-agents sont particulièrement appropriés lorsqu' on s'intéresse aux systèmes complexes. En effet. Ils permettent de reproduire le fonctionnement global d'un système complexe à partir des entités qui le compose et de leurs interactions. [24]

C'est un nouveau niveau d'abstraction qui permet d'exprimer une application en termes d'agents autonomes qui jouent des rôles et rendent des services dans une organisation.

Le développement de l'informatique en termes de puissance des machines, l'évolution des réseaux particulièrement le web et l'augmentation de la masse d'information à traiter et à stocker ont engendré des besoins en applications assez complexes. En général, ces dernières sont physiquement et fonctionnellement distribuées.

Les systèmes multi-agents répondent à ce genre d'application, il, permettent la conception modulaire du système. Les modules peuvent être répartis sur plusieurs machines .un module est plus simple à concevoir qu'un programme monolithique.de plus, la maintenance du système est plus facile, l'amélioration d'un traitement est localisé en général au niveau d'un (ou d'un sous ensemble d') agent (s) l'approche par les systèmes multi-agents propose des solutions robustes et capable de s'adapter dans des environnements qui peuvent être aussi imprévisible que l'internet .Par ailleurs, le problème de ce genre système réside dans la coordination et la gestion des interaction entre les agents qui sont autonome .souvent hétérogène , mais fonctionnant dans un environnement

commun de réaliser une fonction cohérente dans un cadre collectifs et de former ainsi un système .

3.5.3 Caractéristiques d'un SMA

Le SMA est un système et tous les systèmes ont des caractéristiques et des propriétés.

Un SMA a les caractéristiques suivantes : [28]

- **Ouvert** : les agents y entrent et en sortent librement (ex : une application de commerce électronique, etc.).
- **Fermé** : l'ensemble d'agents reste le même.
- **Homogène** : tous les agents sont construits sur le même modèle.
- **Hétérogène** : des agents de modèles différents, de granularités différentes.
- **Mixte (ou non)** : les agents « humains » sont partie intégrante du système, comme le serait un groupe de travail représenté par des agents assistants (implique ouvert et hétérogène)

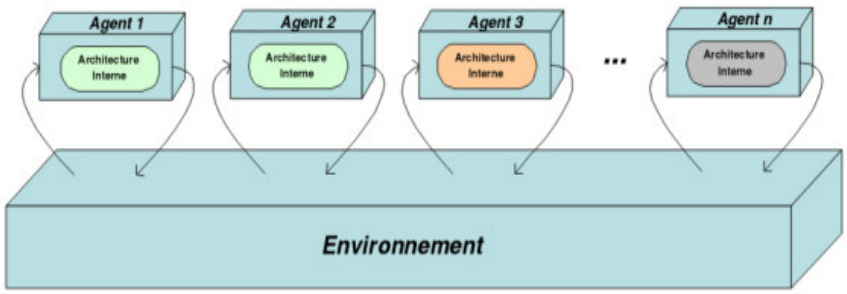

Figure 3.4 : « Architecture d'un SMA » *[28]*

3.5.4 Communication entre les agents

La communication est la base de la résolution coopérative des problèmes. Elle permet de synchroniser les actions des agents et résoudre les conflits de ressources et de buts par la négociation. Dans les systèmes multi-agents, les agents ne disposent d'aucune mémoire commune. La communication entre les agents repose explicitement sur des mécanismes d'envoi de message, de réception et de synchronisation.

3.5.4.1 Architecture de communication

La communication entre les agents dans les SMA peut être organisée suivant trois schémas différents : [29]

- **Réseaux en anneau :** Les interactions dans ce genre d'organisations ont trop lentes, un message destiné à un agent doit transiter par n-1 agents (figure 3.5) :

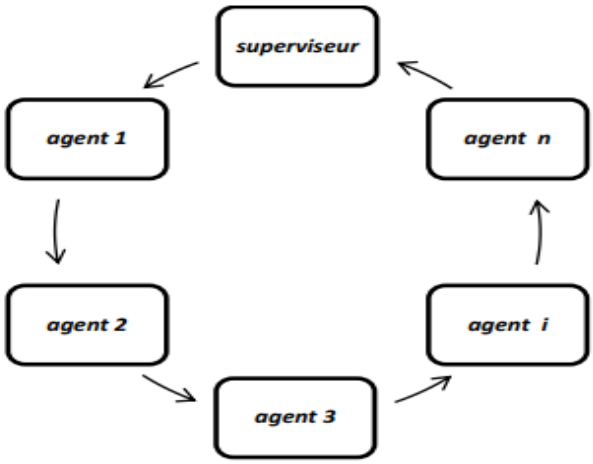

Figure 3.5 : « Réseau en anneau »

- **Réseaux en étoiles** : Ils présentent l'avantage de l'accès rapide entre le superviseur et les autres agents. La nature bidirectionnelle des connexions rend complexe la gestion des interactions inter-agents (figure 3.6). [29]

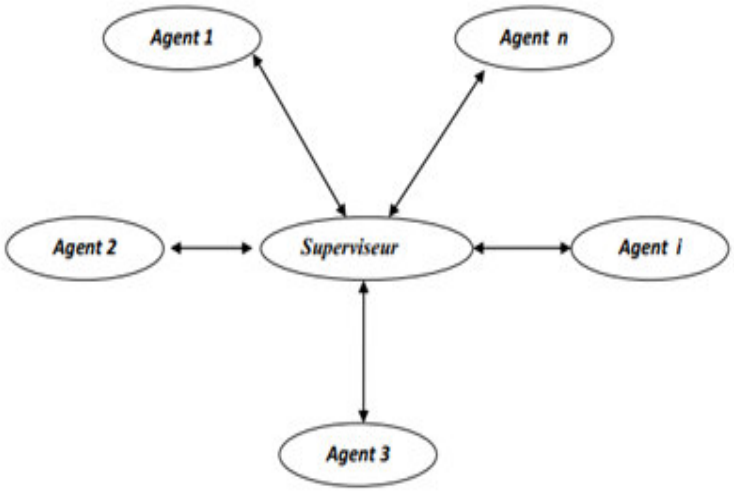

Figure 3.6 : « Réseau en étoile »

❖ **Réseaux hybrides** : Ce type d'organisation est une solution hybride reliant deux types d'organisation : un réseau en bus et un réseau en étoile (figure 3.7).

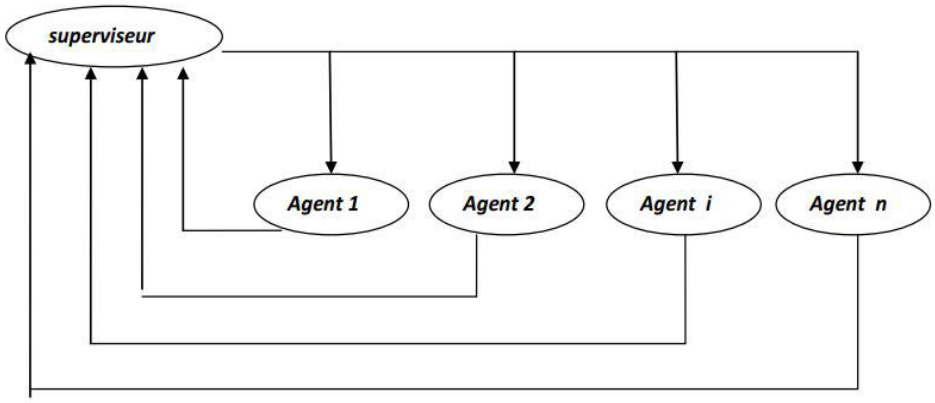

Figure 3.7 : « Réseau hybride »

3.5.4.2 Mode de communication

✓ ***Communication par partage d'informations***

Les composants ne sont pas en liaison directe mais communiquent via une structure de données partagée, ou on trouve les connaissances relatives à la résolution (état courant du problème) qui évolue durant le processus d'exécution. Cette manière de communiquer est l'une des plus utilisées dans la conception des systèmes multi-experts.

L'exemple parfait d'utilisation de ce mode de communication est l'architecture de blackboard (tableau noir), on parle plutôt de sources de connaissances que d'agents. Ce mode de communication n'existe pas dans les systèmes multi-agent où l'on dispose que d'une vision partielle du système alors que la communication par partage d'informations suppose l'existence d'une base partagée sur laquelle les composants viennent lire et écrire. [29]

- **Communication via le tableau noir**

L'architecture tableau noir se compose des trois éléments suivants :

- Les connaissances.
- Le tableau noir.
- Le mécanisme de contrôle.

Dans cette architecture la communication n'est pas directe et l'interaction entre agents se fait via le partage d'un même espace de travail (figure 3.8)

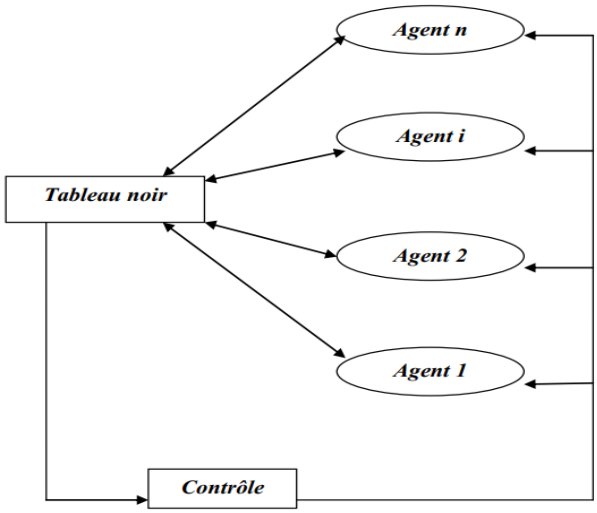

Figure 3.8 : « Architecture tableau noir »

✓ ***Communication par envoi de messages :***

Les agents sont en liaison directe et envoient leurs messages directement et explicitement au destinataire. La seule contrainte est la connaissance de l'agent destinataire. Les systèmes fondés sur la communication par envoi de messages relèvent d'une distribution totale à la fois de la connaissance, des résultats et des méthodes utilisées pour la résolution du problème. [29]

- **Architecture à contrôle distribué :** Distribution totale des connaissances et du contrôle (figure 3.9).

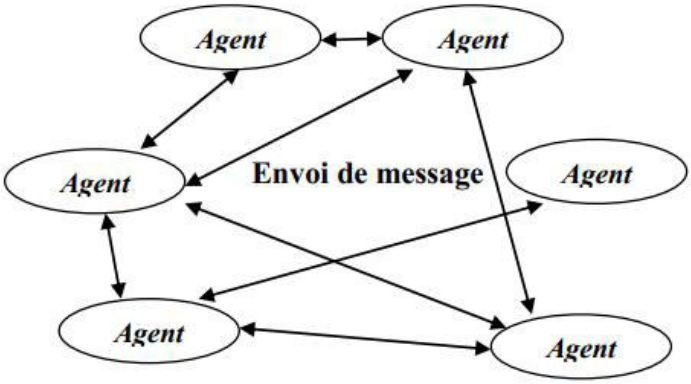

Figure 3.9 : « Architecture à contrôle distribué »

3.5.4.3 langages de communication

Lorsque la notion de groupe d'agents est apparue, les recherches sur des mécanismes permettant aux agents de communiquer entre eux se sont amorcées. Celles-ci ont mené à trois langages de communication et de représentation de l'information qui sont devenus des « standards » en SMA :

KQML (Knowledge Query Manipulation Language). KQML détermine un format pour les messages et un protocole pour la réception et l'envoi de ces derniers. KQML permet aux agents de partager des informations avec les autres agents du système afin de coopérer pour résoudre un problème [29].

Il y a d'autre langage qui s'appelle Le langage **(FIPA ACL**), est un langage semblable à KQML auquel des protocoles d'interaction comme le contractent et autre protocole populaire ont été ajoutés.

3.5.4.4 Interaction entre agents

Ferber définit les interactions comme étant la mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques. L'interaction entre agents s'effectue par la communication, les actes de langages et les protocoles d'interaction. Les agents interagissent entre eux. Pour atteindre son objectif ou pour améliorer la coordination

des actions, un agent peut demander des services à un autre agent. TI y a deux éléments qualitatifs qui caractérisent les interactions entre agents : [21]

- Les interactions se produisent à un niveau élevé par un langage de communication, en fonction du temps, de l'objectif à atteindre et de la nature des agents ;
- Comme les agents sont des solutionneurs de problèmes flexibles dans un environnement au-delà duquel ils ont uniquement un contrôle partiel, les interactions devraient également être flexibles. Ainsi, les agents ont besoin d'un appareil de calcul pour prendre des décisions, dans un contexte dépendant de certains facteurs, au sujet de la nature et de la portée de leurs interactions et pour provoquer des interactions qui n'ont pas été prévues lors de la conception.

Les interactions définissent un contexte organisationnel qui définit la nature des rapports entre agents et qui influence le comportement individuel des agents. TI est donc important de représenter explicitement ces rapports qui sont sujets à un changement continu. Les rapports existants évoluent dans le temps et de nouvelles relations se créent.

Pour faire face à cette variété et cette forme dynamique de rapports, les chercheurs ont :

- Conçu des protocoles rendant capable la formation du contexte organisationnel ;
- Spécifié des mécanismes permettant aux agents d'agir ensemble de façon cohérente ;
- Ils ont également développé des structures pour caractériser le macro-comportement de l'ensemble.

3.6 Domaine d'application

Les systèmes multi-agents ont déjà montré des promesses particulières dans une variété de systèmes technico-sociaux comme : la gestion du trafic aérien, la gestion des ressources, les applications boursières, la télémédecine, l'environnement de robotique cognitive ou les systèmes de commande et de contrôle en temps réel, etc. [30]

Aussi, la technologie agent a trouvé sa place dans les systèmes dynamiques, où l'action autonome, la flexibilité et l'intelligence sont requises quand les quantités de données à traiter dépassent l'envergure de traitement sur une seule machine.

Cette approche constitue un bon moyen de résolution et d'appréhension des problèmes. Cela, explique l'utilisation actuelle des systèmes multi-agents dans divers domaines, dont nous avons cité quelques-uns brièvement :

- Le web sémantique et la gestion documentaire sur le web ;
- Les réseaux tels que Peer to Peer (P2P) ;
- Les systèmes distribués concurrents et les bases de données reparties ;
- Le commerce électronique sur le net et la gestion des processus d'affaires ;
- Les systèmes d'information coopératifs ;
- Les télécommunications.

3.7 Plates-formes multi-agents

Des plates-formes typiques incluent l'architecture d'un ordinateur, le système d'exploitation, programmant des langues et l'interface utilisateur liée (des bibliothèques de système de temps d'exécution ou l'interface utilisateur graphique). [28]

Une plate-forme est un élément crucial dans le développement logiciel. Une plate-forme pourrait être simplement définie comme un endroit pour lancer le logiciel. Le fournisseur de plate-forme offre un engagement au développeur de logiciels que le code de logique exécutera successivement tant que la plate-forme fonctionne en plus d'autres plates-formes.

- **JADE**

(Java Agent DEvelopement) est un Framework de développement de systèmes multi agents, open-source et basé sur le langage Java. Jade fournit des classes qui implémentent « JESS » pour la définition du comportement des agents.

L'outil possède trois modules principaux (nécessaires aux normes FIPA). Le DF « Director Facilitor » fourni un service de pages jaunes à la plate-forme. L'ACC « Agent Communication Chanel » gère la communication entre les agents. L'AMS « Agent Management System » supervise l'enregistrement des agents, leur authentification, leur accès et utilisation du système. Les agents communiquent par le langage FIPA ACL. Un éditeur est disponible pour l'enregistrement et la gestion des agents. Aucune autre interface n'est disponible pour le développement ou l'implémentation.

- **CORMAS**

(COmmon Resources Multi-Agent System) est un Framework de développement de systèmes multi-agents, open-source et basé sur le langage de programmation orientée objet Small Talk.

- **MAGIQUE**

Est une plate-forme pour agents physiquement distribués écrite en Java et fournissant un modèle de communication original d'appel à la cantonade. Dans MAGIQUE, les compétences sont dissociées des agents. L'architecture des agents et les différentes compétences sont développées séparément. Les compétences sont ensuite greffées comme plugin dans les agents au gré du concepteur.

3.8 L'adaptation et le SMA

Lors de l'étude des systèmes multi-agents, l'adaptation est un aspect qui est considéré comme important. Cet aspect est particulièrement mis en avant par D. Leste! Et il est très apparent lorsque 1'on observe les comportements des agents développant une auto organisation du fait de leurs multiples interactions. L'adaptation est alors une conséquence des phénomènes auto-organisateurs dans un tel système. [29]

Selon J. Ferber, l'adaptation peut se voir sous deux aspects : la forme structurelle et la forme comportementale. L'adaptation structurelle concerne des mécanismes collectifs, comme L'auto-organisation ou la reproduction, ce qui amène à considérer ce type d'adaptation comme une forme d'évolution. L'adaptation comportementale concerne l'agent en tant qu'individu et s'apparente à une accumulation d'expérience, ce qui revient à un apprentissage .Ces deux aspects ne sont bien sûr pas nécessairement indépendants, et peuvent chacun avoir leur impact sur l'adaptation globale d'un système multi-agents dans son Environnement

3.9 SMA et les services Web

On assiste aujourd'hui à un rapprochement entre les recherches en SMA et les services Web. Ce rapprochement se manifeste sous différents aspects. Il y a essentiellement deux aspects. [27].

D'une part l'utilisation des agents en tant que cadre conceptuel pour mettre en place des services Web sophistiques, ou encore des outils de planification, d´découverte et

composition des services Web, ce qui est le cas de notre cadre d'´étude, i. e, utiliser les SMAs comme environnement pour la d´découverte des services Web

- D'autre part l'utilisation des services Web comme un cadre architectural et/ou technologique pour mettre en place des SMAs ou des agents accessibles à travers Internet.

L'objectif principal est de rendre les capacités des agents accessibles à travers des services Web.

Le tableau suivant représente une comparaison entre les différentes technologies (services Web et SMA)

Environnement	SMA	Service Web
Description de service	D'description d'agent	WSDL
Registration	DF/AMS	UDDI
Communication	ACL	SOAP
Langage Sémantique	SL	/
Interaction	IP	WS-BPEL WS-CDL
Autre capacités	Négociation	/

Table 3.1 : « Comparaison entre les services web et les SMAs »

La technologie des services web actuel présente quelques limites, nous allons citer quelques-unes.

- Les services web sont passifs jusqu''à ce qu'ils soient invoques.
- Un service web a seulement connaissance de lui-même, mais pas de ses applications ou Utilisateurs clients.
- Un service web n'est pas adaptable, et il n'est pas capable de bénéficier des nouvelles capacités de l'environnement afin d'apporter des services améliorés.
- L'autonomie, l'adaptation et la coopération, points faibles des services web, sont par

ailleurs des domaines qui ont largement ´été explorent par la recherche dans les systèmes multi-agents.

Les SMA peuvent être considérer comme solution aux limites actuelles que présente des Services web car, les services web sont souvent des interfaces de programmes orientés objet (D´enveloppés via J2EE ou .NET) traites pour générer des descriptions WSDL afin de pouvoir communiquer avec des messages SOAP. Ils bénéficient alors des caractéristiques d'abstraction du paradigme orienté objet (encapsulation, héritage, passage de messages, etc.). L'une des majeures ´évolutions consiste `à passer d'une approche orienté objet des services vers une approche orienté agents. Les services tiendraient alors compte du contexte d'ouverture et de distribution de l'environnement, bénéficieraient des caractéristiques de proactivité, flexibilité et d'adaptabilité des agents, et deviendraient de plus réellement interactifs. En outre, comme les agents peuvent potentiellement participer `a des interactions complexes tout en maintenant la coordination avec les autres agents, les services pourraient ˆêtre dotes des prérequis pour la modélisation d'une collaboration dynamique entre services.

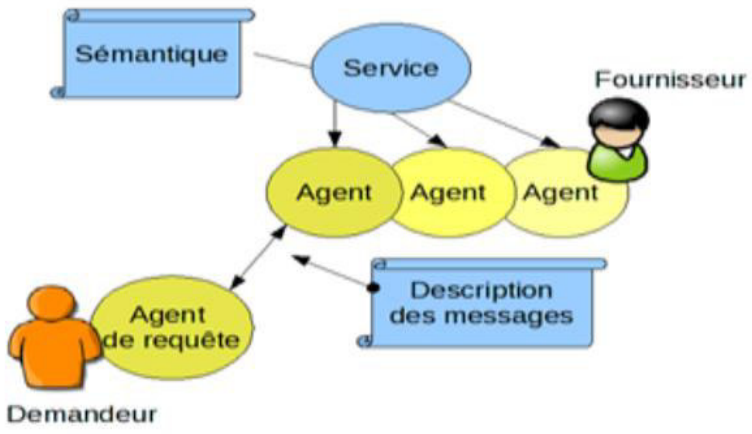

Figure 3.10 : «Les SMA et les services Web»

3.10 Avantages des agents et les systèmes multi agents

L'utilisation des systèmes multi-agents (SMA) est quelque fois obligatoire, quand on se trouve face à des problèmes qui imposent un tel choix. Cependant, nous pouvons choisir d'utiliser des SMA pour les raisons suivantes : [30]

- La majorité des problèmes qui sont distribués et s'adapte facilement aux SMA.

- Les SMA peuvent avoir parmi ces agents celles qui constituants une grande diversité, ce qui donne la possibilité aux concepteurs d'intégrer différents agents (réactifs, cognitifs...etc.).
- Les systèmes peuvent coopérer entre eux pour la résolution de problèmes plus complexes.
- La modularité permet de rendre la programmation plus simple. Elle permet, de plus, aux systèmes multi agents d'être facilement extensibles, parce qu'il est plus facile d'ajouter de nouveaux agents à un système multi agent que d'ajouter de nouvelles capacités à un système monolithique.
- La vitesse est principalement due au parallélisme, car plusieurs agents peuvent travailler en même temps pour la résolution d'un problème.
- La fiabilité peut être également atteinte, dans la mesure où le contrôle et les responsabilités étant partagés entre les différents agents, le système peut tolérer la défaillance d'un ou de plusieurs agents. Si une seule entité contrôle tout, alors une seule défaillance de cette entité fera en sorte que tout le système tombera en panne.
- La facilité de « changement d'échelle d'une architecture » (scalability) car plusieurs agents peuvent s'ajouter ou se retirer dynamiquement d'un système.
- La configuration automatique d'un SMA grâce à la plus grande « proximité » du monde réel des agents.
- La réduction des coûts de développement de logiciel : plus un logiciel n'est modulaire, plus la complexité et les coûts de développement diminuent.

3.11 Inconvénients

Malgré tous les avantages des SMA ils ont quelques inconvénients tels que : [30]

- La sécurité des applications : les agents peuvent communiquer sans aucun contrôle.
- L'exécution des agents s'effectuant en parallèle, il est encore plus difficile de comprendre leur fonctionnement à partir de leur code, du fait du non déterminisme inhérent au parallélisme.
- Enfin, et surtout, les systèmes multi-agents sont des logiciels complexes, difficiles à appréhender et à concevoir. Il est donc nécessaire de réduire leur complexité en dégageant leur structure et en analysant séparément leurs

différents composants sans pour autant perdre de vue l'organisation générale.

3.12 Conclusion

Dans ce chapitre, nous avons présenté une vue générale sur le domaine des Systèmes multi-agents. Le concept de base de ce domaine est la notion d'agent, qui représente une entité autonome capable de percevoir, de se représenter et d'agir sur son environnement.

Hormis les avantages déjà cités dans ce chapitre, les caractéristiques des SMA permettent de respecter les normes du génie logiciel dans l'élaboration des systèmes, à savoir : la modularité, la fiabilité et la réutilisation ce qui explique l'utilisation croissante des SMA dans les systèmes informatiques.

On peut dire aussi que les systèmes multi-agents se trouvent à la croisée de nombreux autres domaines tels que l'Intelligence Artificielle Distribuée, la théorie des actes de langages, mais aussi la théorie des organisations et bien évidemment le génie logiciel, ce qui rend ce domaine un vaste domaine de recherche.

4.1 Introduction

Après avoir présenté dans les chapitres précédents un état de l'art sur les principes de base liés à notre projet, nous allons consacrer ce chapitre à la description de notre proposition du système d'adaptation de service web.

Notre objectif est de proposer une architecture basée agent pour l'adaptation des services web selon le profile utilisateur et ses préférences ; et pour concevoir ce système nous allons étudier deux aspect :

- L'aspect Agent : qui représente l'environnement d'interaction entre agents,
- L'aspect Utilisateur : qui représente l'utilisateur, son profile et ses préférences.

Mais avant de commencer la description de notre système, nous allons définir quelques concepts que nous allons utiliser pour bien illustrer notre architecture.

4.2 Objectif du système

L'objectif de notre système est d'adapter le résultat de l'invocation de service Web en ajoutant ou éliminant des données qui sont jugées pertinentes ou non respectivement. La détermination de la pertinence des données est reliée aux informations de l'utilisateur, son profile, ces buts et ses préférences.

Dans notre système d'adaptation des services web, nous allons utiliser l'adaptation dynamique qui effectue les transformations sur la ressource au cours de son utilisation. Un système d'adaptation automatique intercepte les requêtes des utilisateurs et effectue à la volée des transformations suivant les caractéristiques du contexte d'utilisation actuel. Avec l'adaptation dynamique, nous gagnons un temps de développement important puisque la ressource est transformée automatiquement à la demande de l'utilisateur.

Nous ajoutons que notre architecture est générique où l'agent peut filtrer les résultats de n'importe quel service web sans toucher la structure ou le contenu de ce service. Pour la validation de cette architecture nous allons implémenter une étude de cas dans le chapitre prochain où nous allons filtrer les résultats d'un service web d'une boutique.

4.3 Architecture globale du système

L'architecture définis dans ce travail est une architecture basée agent, conçue pour supporter l'adaptation d'un service web, c'est-à-dire, que l'architecture proposée est sensée contenir tous les concepts nécessaires pour assurer toutes les activités liées à l'adaptation.

Cette architecture définit l'ensemble de composants et modules fonctionnels décrits en termes de leurs comportements et interfaces, ainsi que la façon d'interaction de ces composants afin d'accomplir correctement l'objectif d'adaptation donc, une description architecturale est principalement requise pour la spécification de la structure du système. La figure 4.1 présente les spécifications des différents composants, ainsi que les concepts liés à leur fonctionnement.

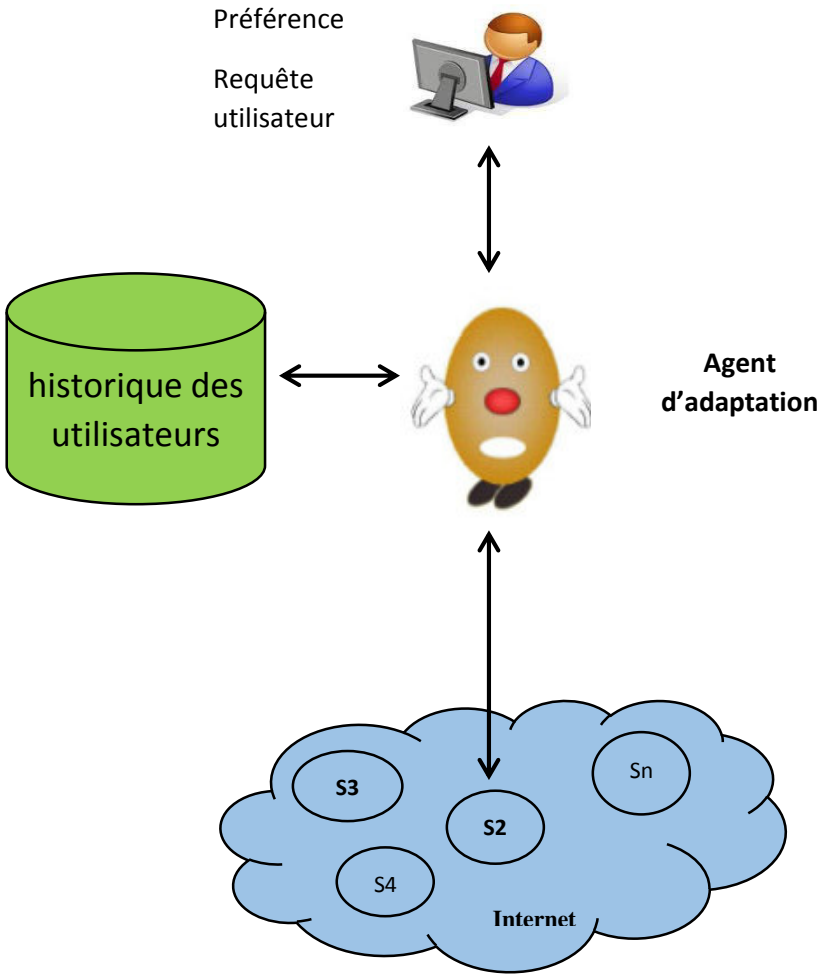

Figure 4.1 : «**Architecture générale du système**»

4.4 Architecture détaillée du système

4.4.1 Architecture d'agent d'adaptation

Comme nous avons mentionné dans chapitre 2, pour adapter une application ou un service web nous devons utiliser le contexte pour réaliser des activités et fournir l'information appropriée aux utilisateurs sans l'interaction humaine. Donc, nous devons assurer une adaptation au type de l'utilisateur connecté pour garantir une utilisation confortable du service web. Pour réaliser cette adaptation, plusieurs paramètres entrent en jeu : paramètres réseau, paramètres de l'utilisateur, paramètres du terminal.

Dans notre architecture le rôle d'agent d'adaptation est d'aider l'utilisateur à identifier ses besoins. Il joue aussi le rôle d'un agent de recommandation en se basant sur l'expérience acquise en interagissant avec les utilisateurs précédents ayant le même profil que l'utilisateur actuel.

Parmi ces paramètres, nous avons choisi les paramètres de l'utilisateur où l'utilisateur est devenu le point central de la conception des systèmes d'information. Nous allons prendre en considération les préférences de l'utilisateur, son emplacement géographique, son profil…

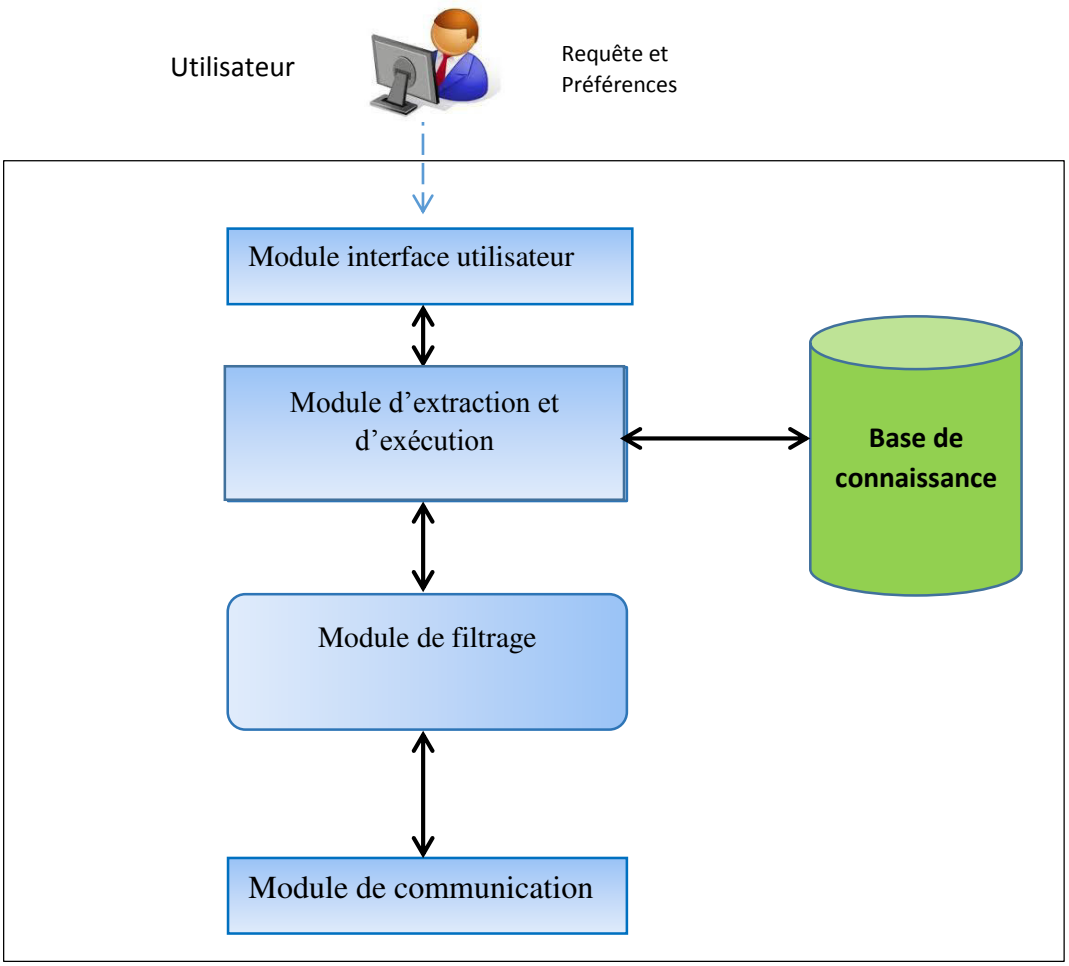

Figure 4.2 : «Architecture d'agent d'adaptation»

1. Module interface utilisateur :

Le module interface utilisateur présente l'interface graphique (homme-machine) du système. L'utilisateur accède via cette interface aux fonctionnalités proposées par le système, et via ce module aussi l'utilisateur peut exprimer ses besoins et ses préférences à l'agent d'adaptation. Nous somme arrivé à choisir le formulaire comme un moyen simple à utiliser par tout le monde et efficace pour exprimer les besoins de l'utilisateur. Une le choix d'utilisateur est fait, l'agent peut établir son profil qui est estimé dans le module d'exécution. Après le traitement le résultat est fourni par le module d'exécution à travers cette même interface.

2. Le module d'extraction et d'exécution

Le rôle de ce module est d'extraire et de définir le profil utilisateur, sachant que les données contenues dans celui-ci sont dépendantes des applications qui l'utilisent. Notre approche est basée sur une méthode simpliste pour la construction d'un profil utilisateur par la construction du profil en utilisant, des mots-clés qui seront fournis par l'utilisateur lui-même. Cette approche est simple car elle fait appel à l'utilisateur à tous les niveaux. En fait, l'utilisateur pourrait trouver assez difficile de fournir les mots-clés qui décrivent convenablement ses préférences. Ainsi, bien que ceci soit une possibilité, nous ne pouvons demander à l'utilisateur de décrire précisément son profil, car ceci serait laborieux. Ce module joue deux rôles :

- Extraction du profile utilisateur : Dans la définition d'un profil, nous distinguons principalement : les données personnelles, les centres d'intérêt, la qualité attendue, les préférences de livraisons, la sécurité et l'historique des interactions de l'utilisateur. La fonction de ce module est de créer un profile utilisateur en se basant sur ses préférences et sa requête. Nous allons présenter le profile sous forme d'un vecteur.

  $$PU=\{ pu_1, pu_2,\ldots, pu_i,\ldots, pu_n]$$

  Sachant que : *PU* : profile utilisateur,

  pu_i : élément i du profile utilisateur,

  n : nombre des éléments du profile utilisateur.

- Reformulation de la réponse pour l'utilisateur : La réponse retournée à l'utilisateur se reformule selon le résultat du filtrage fourni par le module de filtrage.

3. Le module de filtrage

Le module de filtrage est basé sur le profil utilisateur qui représente l'ensemble des informations décrivant l'utilisateur. Il est souvent lié aux concepts de préférences et de contexte.

Ce module permet la fourniture de l'information adéquate au module d'extraction et d'exécution. Il permet l'élimination des données indésirables sur un flux entrant, plutôt que

la recherche de données spécifiques sur ce flux. En utilisant le *profil utilisateur* qui assure de décrire les préférences de l'utilisateur. Un tel profil est alors utilisé afin d'essayer de déterminer les résultats du service web qui pourraient être intéressants pour cet utilisateur particulier. Le service web ne retourne pas à l'agent seulement les résultats pertinentes mais aussi les résultats non-pertinents et le rôle de ce module est filtrer ces résultats. L'avantage du filtrage des résultats est sa simplicité parce qu'il ne nécessite aucune modification du fonctionnement des fournisseurs d'information. Nous pouvons résumer le fonctionnement de ce module selon les trois étapes suivantes :

a. Recevoir une requête utilisateur du module de ré-ordonnancement et d'exécution et l'envoyer au service web qui retourne à son rôle un ensemble de résultats pertinents et non pertinents.
b. Filtrer l'ensemble du résultat du service web en se basant sur chaque élément des éléments du profile utilisateur.
c. Retourner au module de ré-ordonnancement et d'exécution un ensemble de résultat pertinent à cet utilisateur.

4. Le module de communication

Le module de communication est responsable de l'interaction de l'agent avec le service web où il lui permet d'invoquer et de recevoir le résultat du service web.

5. La base de connaissance

Dans ce module est stockée toute la connaissance de l'agent :

- Les informations transmises par l'agent et les profils des utilisateurs.
- Le profil de l'utilisateur actuel.

4.4.2. Le Service Web

Dans notre architecture le service est comme une boite noire où l'agent d'adaptation invoque le service web et reçoit des résultats sans prendre en compte la structure de ce service.

4.4.3. Une base de données

Elle contient l'historique des différents clients et utilisateurs. L'agent d'adaptation stocke l'historique des différents clients dans cette base de données.

4.5 Conclusion

Dans ce chapitre, nous avons présenté une architecture basée agent pour l'adaptation du résultat d'un service web. Cette architecture comporte les concepts nécessaires pour assurer les exigences que nous avons prises en compte afin d'assurer le bon fonctionnement du système proposé. Alors les différents mécanismes d'adaptation sont illustrés, les paramètres d'adaptation sont discutés. Néanmoins, une étude de cas est nécessaire pour l'évaluation des différentes idées dans un environnement réel. Cette étude de cas va permettre d'aborder la phase d'implémentation. Cette dernière phase fera l'objet du chapitre suivant pour la validation de notre approche.

5.1. Introduction

Après la présentation de la conception du notre approche dans le chapitre précédent, nous allons dans ce chapitre décrire les différents aspects techniques liés à l'implémentation et la mise en œuvre de notre système de l'adaptation des services web.

Nous avons fait appel à un ensemble d'outils et de langages de développement permettant la réalisation de notre application, nous tenons à les présenter tous en argumentant nos choix avant d'entamer la description de notre implémentation. Et pour bien illustrer notre approche, nous allons présenter les différentes interfaces ainsi que chacune de leurs fonctionnalités.

5.2 Étude de cas

Dans cette partie, nous présentons un scénario illustrant les objectifs des travaux réalisés dans cette étude.

☞ ***Scénario***

En utilisant son PC chez lui, Ahmed souhaite acheter un t-shirt parmi les t-shirt s disponibles dans les magasins. Cependant, il a trouvé de nombreux services Web fournissant ce service, Mais Ahmed ne veut pas perdre son temps à chercher le t-shirt en vérifiant tous les services Web concernés par son requête. Parce qu'il veut acheter un t-shirt de sa couleur préférée et de sa taille moyenne (couleur rouge, taille M).

Cela signifie qu'Ahmed recherche une solution permettant d'acheter le t-shirt rapidement et en tenant compte de ses préférences.

5.3 Outils de développement du système

5.3.1 Outils matériels 

Pour implémenter notre approche, nous avons utilisé un PC portable HP doté de Windows 7 (64bits), RAM 4Go , CORE i3 ,HDD 500 Go , Figure 5 .1 :

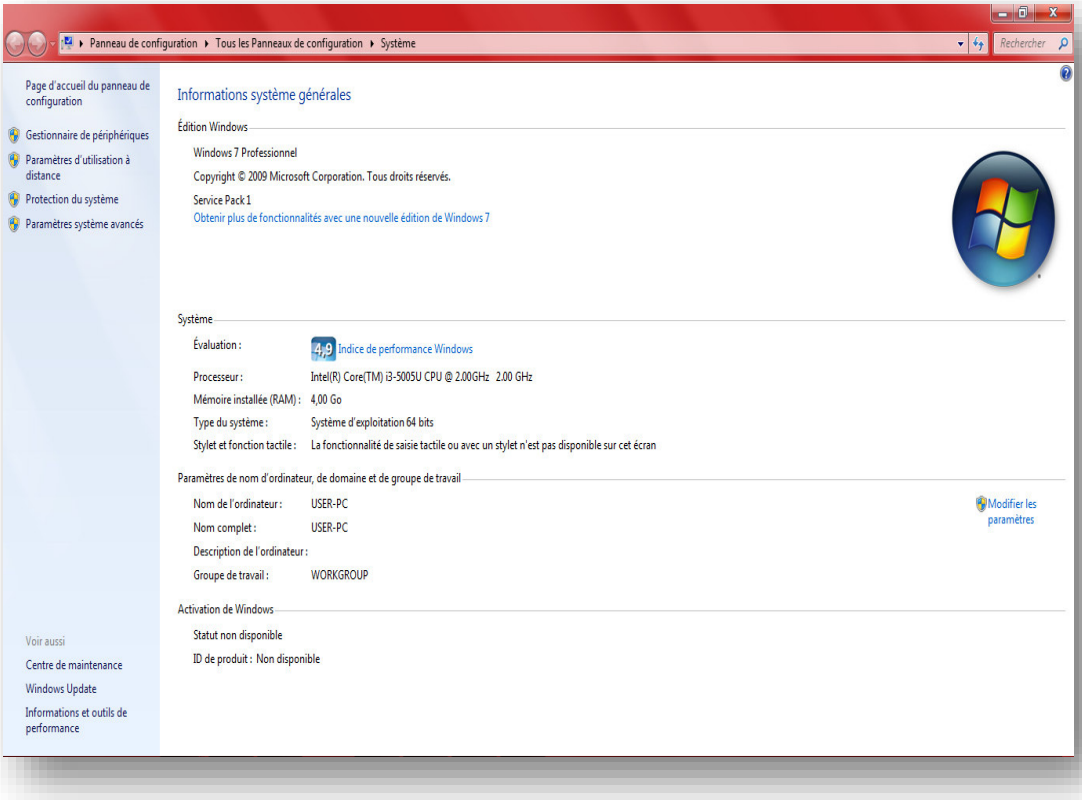

Figure 5.1 : « Information système générale »

5.3.2. Outils de programmation

5.3.2.1 Pourquoi JAVA ?

❖ JAVA est un langage de programmation développé par Sun Microsystème (les premières versions datent de 1995). Il réussit à intéresser beaucoup de développeurs à travers le monde. Java assure les caractéristiques conceptuelles d'un agent ou bien d'un système d'agent. [32]

❖ **Caractéristiques de JAVA**

Java possède plusieurs caractéristiques importantes, en voici quelques-unes :

- *Java est simple* : C'est un langage simple à prendre en main, basé sur le langage C/C++ mais laisse de côté les sources de problèmes (pointeurs, structures, gestion de la mémoire, héritage multiple, macros etc.).
- *Java est orienté objet* :Java est un langage purement orienté objet, tout est classe. Héritage simple. Une librairie plus de classes est fournie.
- *Java est distribuée* : Propose une API réseau standard. Cette dernière permet de manipuler, par exemple, les protocoles http & FTP avec aisance.
- *Java est indépendante de l'architecture* : Le bytecode généré n'est pas lié à un système d'exploitation en particulier. De ce fait, il peut être interprété très facilement sur n'importe quel environnement disposant d'une JVM.
- *Java est portable*
- *Java est robuste :*
  - ✓ Pas de pointeurs.
  - ✓ Gestion de mémoire indépendante.
  - ✓ Mécanisme d'exceptions pour la gestion des erreurs.
  - ✓ Compilateur très contraignant.
  - ✓ Pas d'héritage multiple ni surcharge des opérateurs.
- *Java et sûr*

✓ 4 niveaux de sécurité :

- Langage et son compilateur contraignant.
- Vérifier : vérifier le bytecode.
- Class Loader : le chargeur de classe.
- Security Manager : protection des fichiers et accès au réseau.

- *Java charge dynamiquement les classes suivant les besoins de l'application (pas d'édition de lien).*
- *Java est multithread* : Un Thread est un flot d'instruction s'exécutant en concurrence avec d'autres threads dans un même processus.

❖ JAVA : c'est le langage la plus utilisé dans les plateformes de développement des systèmes multi agents parce qu'il se caractérise par : Simple, Sécurité, portable (Écrire une fois, utiliser partout) et multitâches. [32]

☞ **Nous avons utilisé la version de JDK 1.8.**

5.3.2.2 L'environnement Eclipse 

❖ Pour notre choix de l'environnement de développement Java, nous avons opté sur l'utilisation de L'environnement Eclipse. [33]

- Éclipse est un environnement de modélisation et de développement générique, ouvert et extensible.
- Éclipse est générique puisqu'il permet le développement peu importe le langage utilisé (Java, C/C++, Cobol, XML/XSL, UML...) sur de nombreux systèmes d'exploitation (Linux, Windows, Solaris, QNX, AIX, HP-UX, Mac OSX);
- Éclipse est une plateforme ouverte puisqu'elle est offerte sous licence Common Public

License, c'est-à-dire que le code source est libre de redevance n'importe qui peut le redistribuer et les travaux dérivés sont autorisés ;

- Éclipse est également extensible puisque c'est un Framework permettant de construire et d'intégrer des outils de développement de toute nature.
- Éclipse vient avec un ensemble de modules et de bibliothèques servant à la gestion des ressources. On peut créer des projets, éditer et sauvegarder des fichiers, imprimer, partager des ressources, gérer les versions à l'aide d'une interface CVS intégrée, etc.
- Éclipse offre aussi de façon standard un environnement de développement Java et des outils de développement de modules d'extension.
- L'environnement de développement Java, qui n'est qu'un ensemble de modules ***d'e***xtension (appelé JDT), offre un éditeur spécialisé, une compilation incrémentale, un débogueur et différents services tels que le code complétion, des code Template et le refactoring.
- Ce qui distingue *Eclipse* des autres *IDE* est l'extensibilité de son environnement. *Eclipse* a été conçu de manière à pouvoir facilement étendre ses fonctions à l'aide de modules d'extension tout en conservant une interface graphique cohérente. On Peut ainsi charger différents modules dans *Eclipse* pour le développement en tout genre comme *Java*, *C/C++*,*Cobol*, *XML/XSL*, *UML*, etc.

☞ **La version utilisée est Éclipse 2018-09 (4.9).**

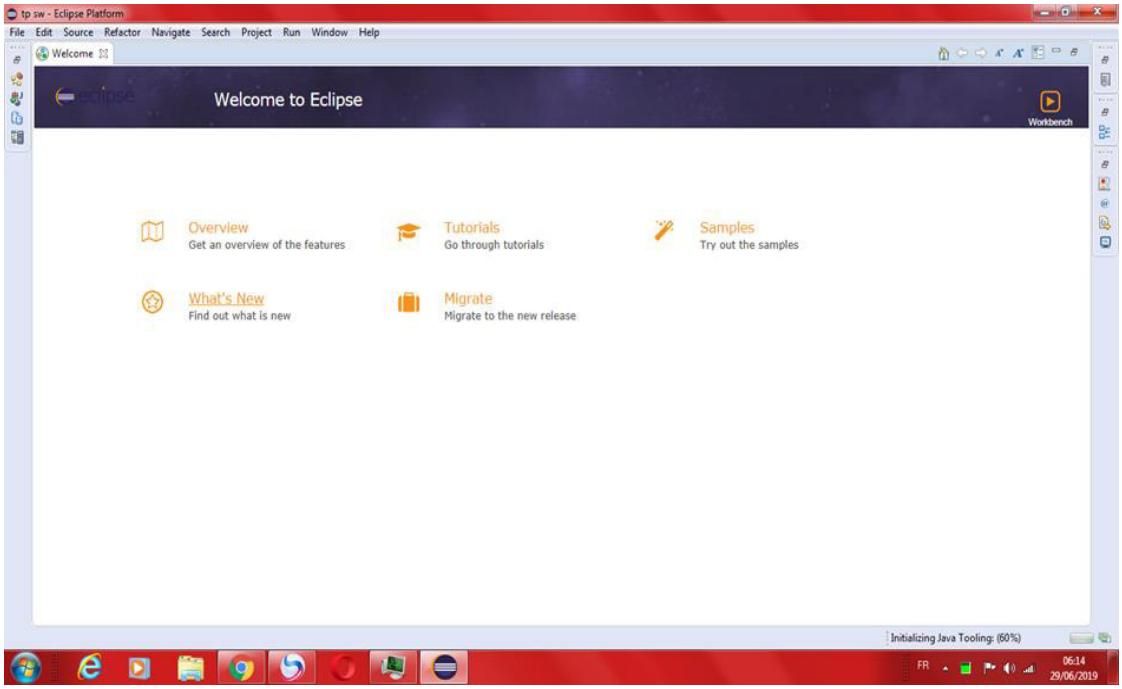

Figure 5.2 : «L'environnement de développement Eclipse version 2018-09 »

5.3.2.3 L'environnement JADE

Afin d'assurer un développement rapide est efficace des agents qui composent notre système, nous utilisons la plateforme de développement des systèmes multi-agent JADE.

JADE regroupe un ensemble d'outils et d'API qui permettent la construction et la mise en service d'agents sur un contexte bien spécifique. [34]

❖ **Présentation générale**

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

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

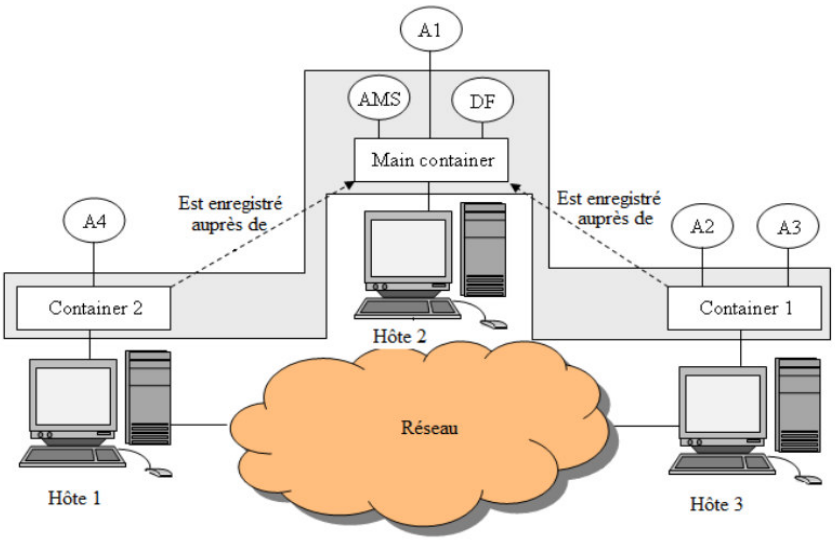

Figure 5.3 : «Architecture de JADE»

❖ **Architecture logicielle**

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

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

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

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

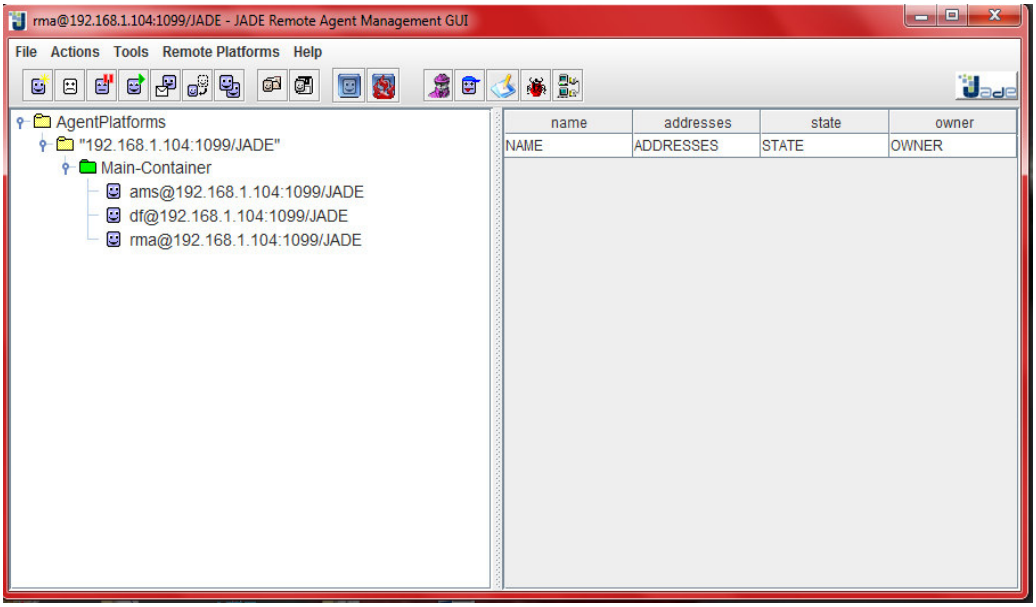

Figure 5.4 : «L'interface graphique de la plateforme JADE»

❖ Langage de communication

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

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

☞ **La version utilisée est *JADE* 4.5.**

5.3.2.4 WebDev et WinDev

Est un atelier de génie logiciel (AGL) édité par la société française PC SOFT et conçu pour développer des applications, principalement orientées données pour Windows et également pour Linux, .NET et Java. Il propose son propre langage : le WLangage. La première version de l'AGL est sortie en 1993.[35]

WebDev et WinDev Mobile permettent d'utiliser le même langage de programmation (WLangage), et les mêmes concepts (analyse, fenêtre, états, composants, classes…), pour la génération de sites Web et d'applications mobiles.

WinDev inclut en standard un ensemble d'éditeurs qui composent l'Atelier de Génie Logiciel : éditeur d'analyse (description des données), éditeur de fenêtres, éditeur de requêtes SQL, éditeur d'états, éditeur de tests automatisés, éditeur d'aide, éditeur d'images, éditeur UML, éditeur de code, éditeur de télémétrie, robot de surveillance, audit d'application, éditeur de dossier…etc[36]

La programmation s'effectue typiquement dans les composants graphiques, en saisissant directement le code dans les événements proposés.

❖ **Serveur d'application WebDev[37]**

WebDev est constitué de :

- **Webdev Version Développement**

  Installé sur le poste de développement, cette version permet de développer un site Webdev ou un service web et de le tester en local

- **Serveur d'application WebDev**

  Installée sur un serveur chez l'hébergeur , cette version permet de déployer un site dynamique WebDev (site avec base de données).

  Le site WebDev peut étre utilisé par tous les internautes.

☞ **Les possibilités du serveur d'application WebDev**

Doit être installé sur un poste serveur (chez l'hébergeur ou sur un poste serveur intranet). Grace au serveur d'application WebDev :

- Les internautes peuvent utiliser des sites dynamiques WebDev .

- L'administrateur du serveur peut
  - Gérer et configurer les différents site dynamique WebDev présents sur le serveur,
  - Configurer les comptes associés a chaque responsable de site
  - Controler l'installation et la mise a jour des sites dynamique à distance (par FTP) ,
  - Surveiller les serveurs où des sites dynamiques WebDev sont installés

☞ **La version utilisée est WebDev 10.**

5.3.3 Outils de base des données

Pour créer et gérer les bases de données nécessaires pour notre système, nous avons utilisé les outils Suivants :

5.3.3.1 WampServer

Est une plateforme de développement Web de type WAMP, permettant de faire fonctionner localement (sans avoir à se connecter à un serveur externe) des scripts PHP. WampServer n'est pas en soi un logiciel, mais un environnement comprenant deux serveurs (Apache et MySQL), un interpréteur de script (PHP), ainsi que phpMyAdmin pour l'administration Web des bases MySQL. Il dispose d'une interface d'administration permettant de gérer et d'administrer ses serveurs au travers d'un tray icon (icône près de l'horloge de indows).[32]

☞ **La version utilisée est WampServer 2.5**

5.3.3.1 MySql

MySql est un Système de Gestion de Base de Données Relationnelles. Avec plusieurs millions de serveurs installés en production dans le monde, MySql est devenu en quelques années le serveur de base de données libre le plus utilisé. Il est compatible avec les principaux standards SQL, MySql a su s'imposer comme la référence base de données des applications Internet, Intranet et Extranet.

Afin de créer une base de données en MySql on a utilisé l'outil phpMyAdmin qui offre une interface intuitive pour l'administration des bases de données MySql[38]

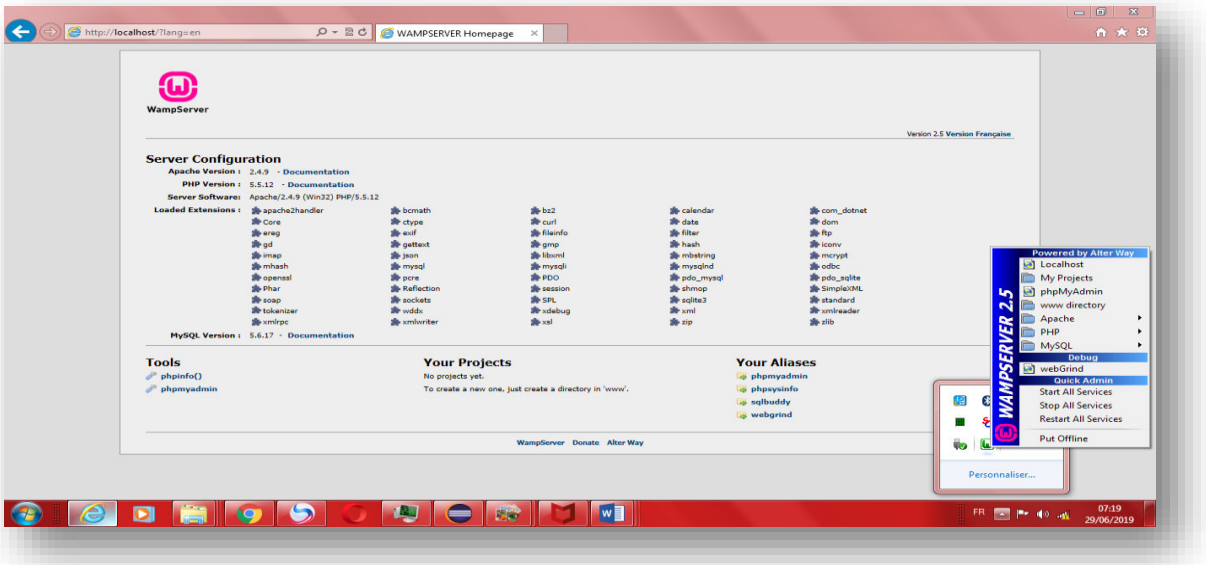

Figure 5.5 : «L'interface graphique de la plateforme WampServer»

5.4 Présentation de l'application

Dans cette section nous détaillons l'environnement d'utilisation du système implanté dans la plateforme JADE et basé sur le système multi-agents.

Comme nous l'avons mentionné dans la section 5.3, l'implantation de l'application de la plateforme JADE a été mise en œuvre en utilisant le langage Java et les outils Eclipse, WampServer, sous environnement Windows.

5.4.1 Présentation de service web utilisée

Comme nous l'avons mentionné dans la section 5.3.2.4 WINDEV et WEBDEV permettent de générer directement des Services Web. Ces Services Web peuvent ensuite être utilisés dans des projets WINDEV, WEBDEV, WINDEV Mobile ou dans n'importe quel autre langage.

Afin de tester l'efficacité de notre système, nous avons créé un service Web utilisant WEBDEV.

Ce Service Web est spécialisé à la vente des t-shirts. (Figure 5.6)

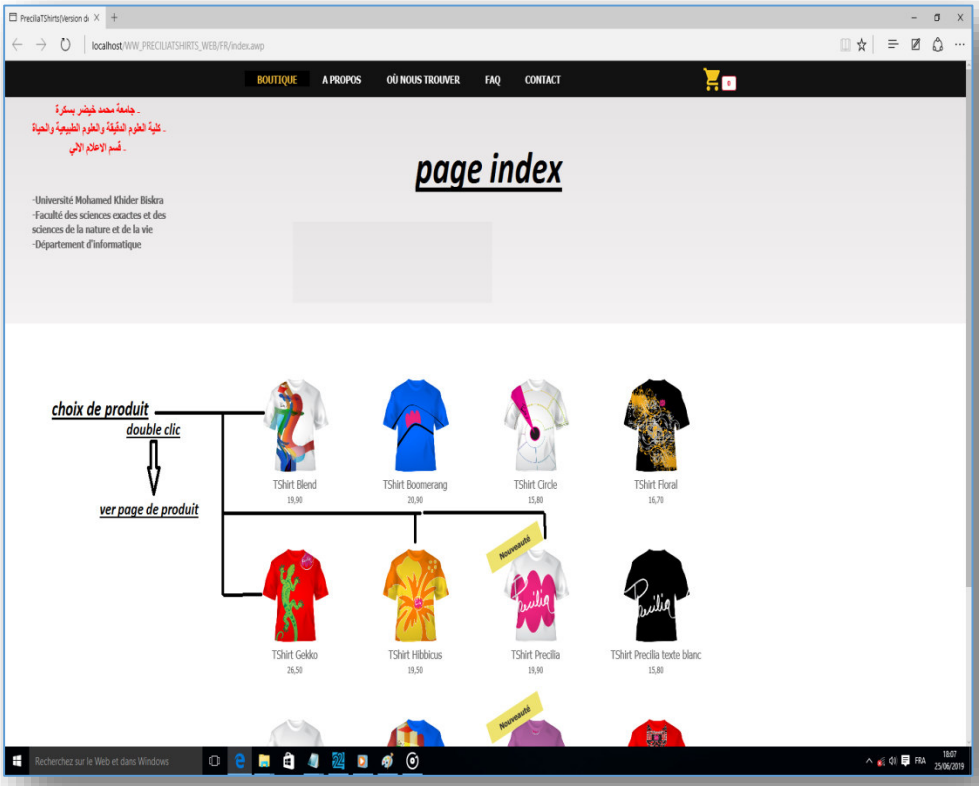

Figure 5.6 : « Page d'accueil de service web ' Boutique t-shirt ' »

5.3.2 Implémentation des agents

Pour décrire l'implémentation des agents, nous allons présenter les différentes classes agents du système sous forme d'un pseudo-code:

Pour créer un agent, on doit définir une classe héritée de la classe « jade.core.agent »

ou « jade.gui.GuiAgent » La deuxième étape est celle d'implémenter la méthode « setup() »où nous spécifions l'ensemble des opérations qui doivent être fait lors de l'initialisation de l'agent.

L'agent utilisateur (module interface utilisateur) interagie avec l'agent adaptateur (formulaire GUI) dans le module d'exécution ; ce dernier interagie avec la Base des sonnées (Base de connaissance) et communique avec l'agent de filtrage (module de filtrage) et les autres agents du système (module communication).

```
import jade . core . Agent ;
public class AgIn extends Agent {
        protected void setup () {

        }
}
```

Figure 5.7 : « Création d'un simple agent»

```
import jade . gui . GuiAgent ;
public class Inter_AgIn extends GuiAgent {
        protected void setup () {

        }
        protected void onGuiEvent ( GuiEvent ev) {

        }
}
```

Figure 5.8 : «Création d'un agent avec interface graphique»

5.4.3 Présentation des interfaces utilisateur

5.4.3.1 l'interface d'accueil

La Figure 5.9 présente une copie d'écran de l'interface principale de l'application proposée. En cliquant dans cette interface, l'utilisateur peut se connecter à notre système.

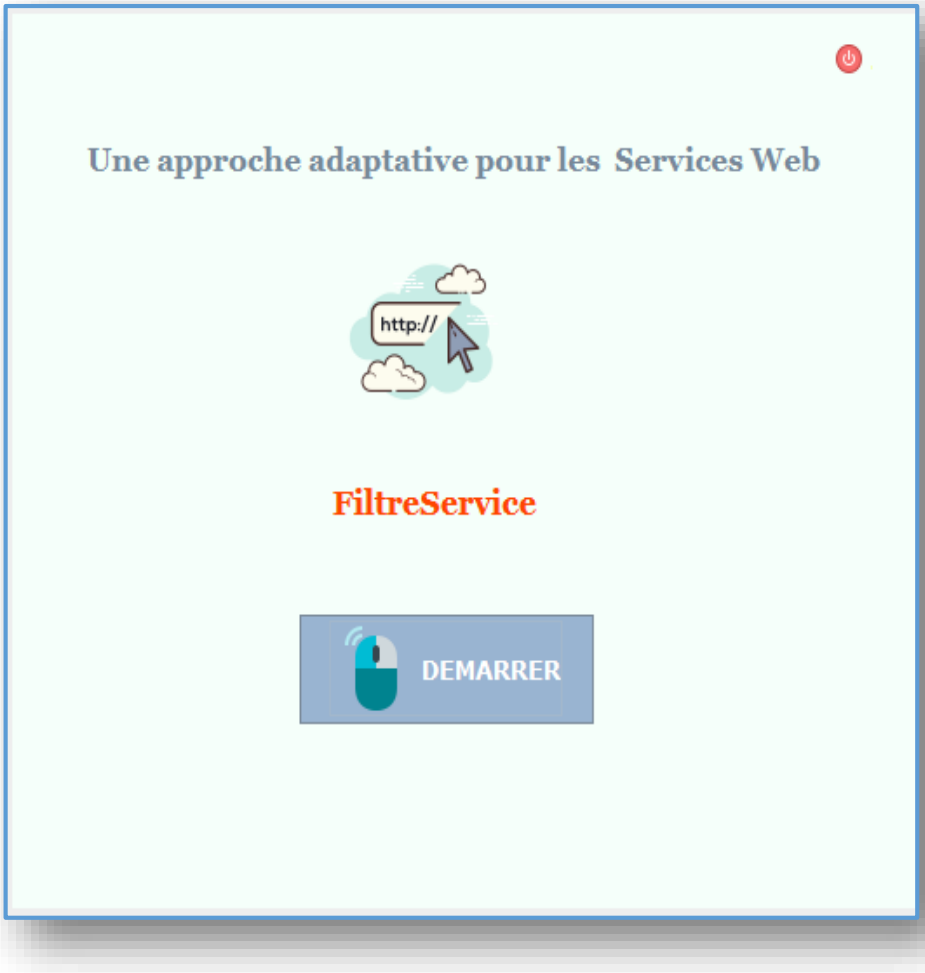

Figure 5.9 : «Interface principale de notre application »

5.4.3.2 l'interface d'ouvrir une session ou créer un compte

Chaque personne se connectant à notre système doit préciser s'il est un ancien utilisateur enregistré ou un nouveau membre souhaite s'inscrire dans le système (Figure 5.10).

Figure 5.10 : «Interface ouvrir une session ou créer un compte »

5.4.3.3L'interface d'authentification

Cette capture présente l'interface d'authentification dans laquelle on doit choisir l'email d'utilisateur et entrer le mot de passe pour commencer à utiliser notre application. Cette interface constitue la fenêtre d'accueil de notre application. A travers cette fenêtre l'utilisateur s'authentifie pour utiliser l'application.

Cette étape met en valeur l'aspect sécurité : nous vérifions la disponibilité du compte utilisateur et nous lui attribuons les droits et privilèges nécessaires.

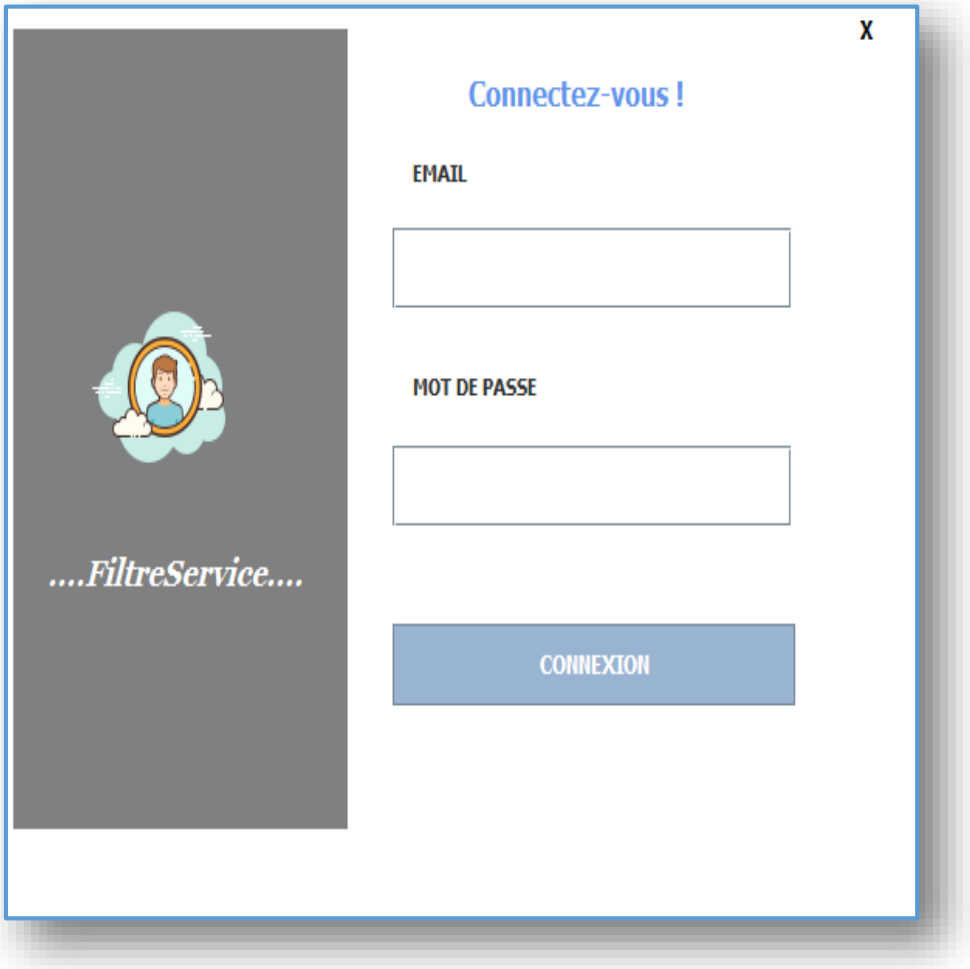

Figure 5.11 : «l'interface d'authentification»

5.4.3.4 L'interface de la Création d'un compte utilisateur

Les informations de base du contexte sont introduites par l'utilisateur lors de son enregistrement. Il enregistre ses informations personnelles (civilité, nom, prénom, date de naissance, numéro du téléphone, profession, et son adresse E-mail).

Figure 5.12 : «l'interface d'un formulaire d'enregistrement»

5.4.3.5 L'interface de Filtrage d'un service web

- Elle permet à l'utilisateur de lancer sa requête dans un service web et obtenir un résultat adapté à son contexte d'utilisation.
- Par exemple, un utilisateur veut acheter un t-shirt et veut connaître les offres proposées par le service Web boutique de t-shirt, Lorsque l'utilisateur contacte notre système, il donne d'abord explicitement ses préférences via une interface (Figure 5.13) en appuyant sur le bouton lancer filtrage, Le résultat de la requête en prenant en compte le contexte apparaît dans la zone 1
- Dans cet exemple, l'utilisateur a précisé deux préférences « par taille» et « par couleur ».
- Les informations concernant le contexte (les préférences) sont enregistrées dans une base de connaissance.
- Après l'affichage du résultat, l'utilisateur choisit un t-shirt de la liste et commence l'interaction avec ce service.

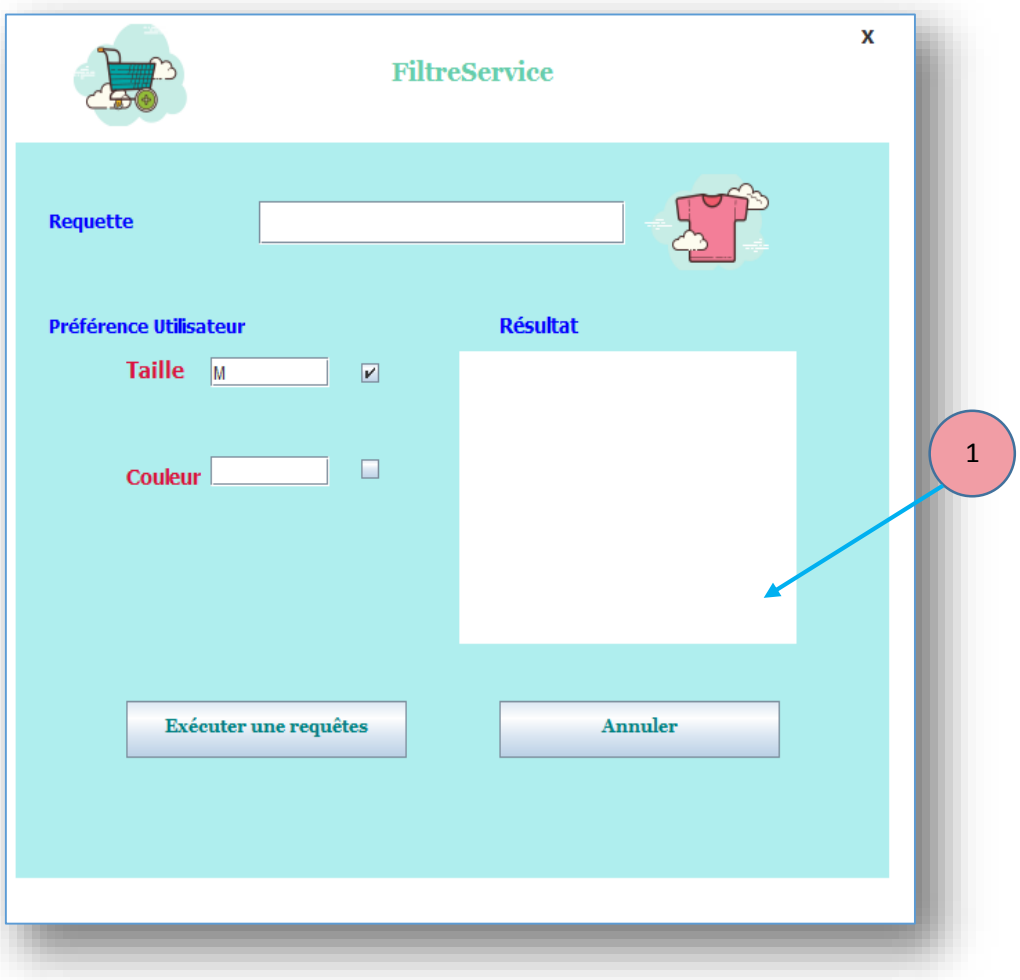

Figure 5.13 : «l'interface de Filtrage d'un service web »

5.4 Conclusion

Au cours de cette dernière étape de notre travail, nous avons réalisé un agent d'adaptation afin de valider notre proposition d'une approche adaptative pour les services web. Nous avons donné une brève présentation des outils utilisés pour l'implémentation, et pour bien illustrer notre implémentation, nous avons fourni une démonstration des différentes fonctionnalités de notre application réalisée à travers des captures d'écran.

Conclusion générale et perspectives

Un des grands besoins liés aux services web est l'adaptation au contexte. Ce besoin vise à délivrer à l'utilisateur des résultats pertinents non seulement par rapport à sa requête mais aussi par rapport à son contexte. L'adaptation au contexte doit donc être prise en compte dans deux phases de l'architecture d'un tel système : la phase de recherche des Services Web et la phase d'interaction avec un Service Web.

Dans notre approche, le dispositif d'utilisateur est responsable de la gestion du processus d'adaptation selon les préférences de cet utilisateur, les caractéristiques de son dispositif, sa localisation, etc. L'intérêt de cette approche est sa très grande flexibilité : parfaitement au courant des conditions effectives d'utilisation (le contexte), le dispositif du client peut choisir les procédures d'adaptation les plus pertinentes.

Le contexte que nous allons utiliser pour adapter le résultat du service web est présenté par le profil utilisateur. Le profil peut contenir des informations brutes, caractéristiques de l'utilisateur (nom, prénom, pseudonyme, adresse, …). Généralement, ce sont des informations qui viennent directement de l'utilisateur, par un procédé de saisie.

Généralement la manière la plus simple d'initialiser un profil est la saisie manuelle des paramètres par l'utilisateur. Il est considéré que l'utilisateur connaît mieux ses exigences et de ce fait, il peut saisir les paramètres dont a besoin le système. Et c'est ce que nous avons utilisé dans notre approche.

Dans notre système d'adaptation de service web, nous avons utilisé l'adaptation dynamique qui effectue les transformations sur les résultats au cours de leur utilisation. Un système d'adaptation automatique intercepte les requêtes des utilisateurs et effectue à la volée des transformations suivant les caractéristiques du profile d'utilisateur actuel. Avec l'adaptation dynamique, nous gagnons un temps de développement important puisque le résultat est transformé automatiquement à la demande de l'utilisateur.

Nous pouvons citer comme perspective pour les chercheurs qui veulent utiliser notre travail où ils peuvent améliorer l'adaptation et la représentation du contexte d'utilisation, par l'introduction des autres paramètres de contexte comme réseau, terminal, services…etc.

~Dédicace ~

À mon pays.

À ma mère, à ma mère, à ma mère et à mon père.

À ma grand-mère pour son encouragement.

À mes frères, mes sœurs et ma famille.

À mes collègues et mes amies.

À tous mes proches.

Fairouz. L

Introduction générale

Á l'inverse de l'informatique traditionnelle qui est orientée traitement, l'informatique orientée services s'intéresse à l'adaptation des services aux besoins de l'utilisateur. Ce besoin vise à délivrer à l'utilisateur des résultats pertinents non seulement par rapport à sa requête mais aussi par rapport à son contexte d'utilisation. Ce contexte pouvant varier à chaque connexion et en cour d'une même connexion. Parmi les paramètres du contexte le profile utilisateur qui présente les données personnelles, les centres d'intérêt, la qualité attendue, les préférences de livraisons, la sécurité et l'historique des interactions de l'utilisateur

L'objectif de ce projet est de restituer à l'utilisateur un résultat pertinent du service web par rapport à son profile. Donc, nous pouvons dire que le but principal de ce travail est d'améliorer la qualité de réponse du service web retournée à l'utilisateur afin qu'il soit satisfait du résultat délivré.

Les agents sont également destinés au développement d'applications distribuées, toutefois, dans une perspective différente aux services web mais complémentaire, les systèmes multi agents offrent un grand intérêt et promettent d'avoir un impact important sur la technologie de services web. Nous Avons utilisé l'agent comme modèle de conception de notre approche.

Ce mémoire est basé sur trois axes principaux : Un état de l'art, Une conception qui présente notre contribution et la validation de notre proposition par l'implémentation d'une étude de cas.

La partie état de l'art est composé de trois chapitres « chapitre 1 : Les Architectures Orientées Services (SOA) et Les services Web » et « chapitre 2 : l'état de l'art qui représente les travaux d'adaptation existants » et « chapitre 3 : Les Système Multi Agents »

Dans le chapitre 1, nous donnons quelques généralités sur l'architecture orientée service SOA et les services web et leurs fonctionnements.

Concernant le chapitre 2, il est consacré à l'adaptation des services web, ses types, ses paramètres et approches.

Dans le chapitre 3, nous définissons quelques concepts sur les agents. Ainsi que les systèmes multi agents.

La deuxième partie, qui contient notre contribution, comporte deux chapitres « chapitre 4 : Conception » où nous allons illustrer le principe de notre approche d'adaptation sur trois volets : description de l'architecture générale, l'architecture d'agent d'adaptation et les modules utilisés.

Et « chapitre 5 : implémentation » présente l'implémentation de notre approche d'adaptation.

Enfin, notre travail se termine par une conclusion générale résumant les grands points qui ont été abordés ainsi que les perspectives futures de notre travail.

LISTE DES FIGURES

LISTE DES TABLEAUX

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : GLSD10/M2/2019

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Génie logiciel et systèmes distribués**

Une approche adaptative pour les Services Web

Par :

LEFGUI Fairouz

Soutenu le 07 juillet 2019, devant le jury composé de :

MOHAMMEDI Amira	MAA	Président
SAHLI Sihem	MAA	Rapporteur
ALOUI Ahmed	MAA	Examinateur

RÉFÉRENCES BIBLIOGRAPHIQUES

[01] : Amina BEKKOUCHE, «**Composition des Services Web Sémantiques À base d'Algorithmes Génétiques**», Mémoire Pour l'obtention du diplôme de Magistère en Informatique, Université Abou-bekr Belkaid Tlemcen , 2012 , pp.19-38.

[02] : Lamia CHAIF, «**Conception et Réalisation d'une Application basée sur l'Architecture SOA : Traitement des Appels d'Offres»**, Mémoire de fin d'études pour l'obtention du diplôme de Master en Informatique, Université Abou Bakr Belkaid– Tlemcen, 2013, p.15.

[03] : KEBIR Khadidja**, « Conception et Réalisation d'une Application basée sur l'Architecture SOA »**, Mémoire pour l'obtention du diplôme de Master en Informatique, Université Abou Bakr Belkaid– Tlemcen, 2017, pp.15-27.

[04] : Céline LOPEZ-VELASCO, **« Sélection et composition de services Web pour la génération d'applications adaptées au contexte d'utilisation»,** thèse pour obtenir le grade de Docteur De L'université Joseph Fourier, Université Joseph-Fourier - Grenoble I, 2008, p.13.

[05] : Youness LEMRABET, **« Proposition d'une méthode de spécification d'une architecture orientée services dirigée par le métier dans le cadre d'une collaboration inter-organisationnelle »**, These présentée en vue d'obtenir le grade de Docteur, Université Lille Nord-de-France, 2012, pp. 38-47.

[06] : ZEKRI Oumia, **« Conception et réalisation d'un modeleur de composition de service web SOAP et REST»**, Mémoire Présenté pour obtenir le diplôme de master académique en Informatique, Université Mohamed Khider – Biskra, 2014, p 6.

[07] : Iheb ABDELLATIF, **« vers une démarche d'aide à la décision pour l'identification des services d'une architecture orientée services »,** Mémoire pour L'obtention De La Maîtrise En Technologies De L'information M. ING., Université Du Québec, 2011, pp.16-22.

[08] : Sofiane Chemaa, **« Une approche de composition de services Web à l'aide des Réseaux de Petri orientés objet»,** Thèse de Doctorat en informatique, université Abdelhamid mehri, Constantine 2, 2014, pp.7-20.

[09] : BENGERMIKH Abdelkader et BENMAMMAR Mohamed, **« La découverte des services web à l'aide de l'algorithme de vote de Coombs »** , Mémoire de fin d'études Pour l'obtention du diplôme de Master en Informatique, Université Abou Bakr Belkaid– Tlemcen, 2018, pp.20 -21 .

[10] : DALI YAHIA Mohamed, « **sélection de service web à base de QoWS** », Mémoire Pour l'obtention du diplôme de Master 2 en Informatique, Université Abou Bakr Belkaid– Tlemcen, 2011, p11.

[11] : Ibrahim A. KEITA, **« Sécurité des services web : restauration d'un message soap après détection d'une attaque par enveloppement sur un élément signé »,** Mémoire pour l'obtention du grade de maître ès science (M.Sc.), Université du québec en Outaouais, 2010, p 31.

[12] : **https://www.loukam.net .**

[13] : Fayçal ABOUZAID, « **Analyse Formelle D'orchestrations De Services Web ».** Thèse présentée en vue de l'obtention du diplôme de philosophiæ doctor, Université Abdelhamid Mehri, Constantine 2, 2010, p 26.

[14] : Aniss ALKAMARI, « **Composition de services web par appariement de signature ».** Mémoire présenté comme exigence partielle de la maîtrise en informatique, université du Québec a Montréal, 2008, p 45.

[15] :HAMECHE Sara Et DJOUHRI Nawal,**« composition De Services Web Abase d'automates hiérarchiques avec la méthode B-Événementiel »**. Mémoire de fin d'études pour l'obtention du diplôme de Master en Informatique, Université A. Mira de Béjaia, **2013**, p 15.

[16] : SOUKKARIEH Bouchra, **«Technique de l'internet et ses langages : vers un système d'information Web restituant des services Web sensibles au contexte**», En vue de l'obtention du Doctorat De L'université De Toulouse, l'Université Toulouse III - Paul Sabatier, 2010, pp.56-89.

[17] : Sonia ACHIRI, **«Une Architecture Agents pour l'Adaptation au Contexte des Systèmes d'Information Ubiquitaires»**, Mémoire Pour l'obtention du diplôme de Magistère en Informatique, Universite Badji Mokhtar-Annaba, 2012, pp.60-73.

[18] : Ahmed BALI**, «adaptation des services web sensible au contexte »,** thèse Présentée pour l'obtention du diplôme de Doctorat en sciences, université Mohamed Kheider Biskra, 2017 , pp.4-31

[19] : Lamia BACHIRI, **«Une approche basée agents pour l'adaptation des services Web»,** Mémoire de fin d'études Pour l'obtention du diplôme de Master en Informatique, université Mohamed Kheider Biskra, 2016 , p 4.

[20] : séminaire VSST (**Veille Stratégique Scientifique et Économique**) a eu lieu à Nancy (INIST) les 30 et 31 mars 2009.

[21] : Arsène SABAS, «**Une Analyse Comparative Des Méthodologies De Développement Vers La Convergence Des Méthodologies De Développement Et La Standardisation Des Plateformes SMA**», Mémoire Présenté Comme exigence partielle de la Maîtrise en mathématiques et informatique appliquées, l'Université du Québec à Trois-Rivières, 2001, pp.2-9.

[22] : ELANDALOUSSI Sid Ahmed, « **développement d'un WEB-MAS pour la conception et fabrication assistées par ordinateur : application à un atelier de pièce mécanique**». Mémoire pour l'obtenir le diplôme de magister, université d'Oran, 2013, pp.21-27.

[23] : Abderrahim SIAM, « **Approche basée Agents Mobiles et Composants pour développer des applications ouvertes et adaptables. Thèse Pour obtenir le diplôme de Doctorat en Sciences»**, Université Constantine 2 Mahri Abdelhamid, 2015, p 51.

[24] : Nachet BAKHTA, **«Modèle Multi-Agent pour la conception des systèmes d'aide à la décision collective»**, Thèse de doctorat en informatique, université d'Oran, 2014, pp.39-45.

[25] : MOUSSAOUI Mohamed Lamine, **« Système Multi Agent pour planification de la production»,** Mémoire pour l'obtenir le diplôme de magister, Université de Djilali BOUNAÂMA Khemis Miliana, 2016, pp.7-9.

[26] : Inaya LAHOUD, **«Un système multi-agents pour la gestion des connaissances hétérogènes et distribuées»**. Thèse de doctorat, Université de Technologie de Belfort-Montbéliard, 2013, p 35.

[27] : Ali YACHIR, **« Modélisation Et Composition De Services Sensibles Au Contexte»**. Mémoire de Magistère en Informatique, Université Abderrahmane Mira de Béjaïa, 2007, p 83.

[28] : GUERMOUDI Mohammed Amine et BENAMAR Abdeladim, **« Conception et Implémentation d'un Système multi-agent Pour le test de primalité de nombre premier».** Mémoire de fin d'études pour l'obtention du diplôme de Licence en Informatique, Université Abou Bakr Belkaid– Tlemcen, 2011, pp. 24 -30

[29] : Alexandre GROULS, **« Agents et systèmes multi-agents : vers une synthèse de ces concepts»**, Mémoire Présenté Comme exigence partielle de la maitrise de la maîtrise en informatique, Université Du Québec A Montréal, 2013, p 50.

[30] : KADDOUR Halima, **«Mise en oeuvre d'une application SMA Dans les réseaux P2P (Supervision system) »**, Mémoire de fin d'études Pour l'obtention du diplôme de Master en Informatiques, Université Abou Bakr Belkaid– Tlemcen, 2015, p 16.

[31] : ELANDALOUSSI Sid Ahmed, **« développement d'un WEB-MAS pour la conception et fabrication assistées par ordinateur : application à un atelier de pièce mécanique»** . Mémoire pour l'obtenir le diplôme de magister, université d'Oran, 2013, pp 31-32.

[32] : AL MUTAWAKEL abdallah, **«Une approche pour la planification dans les systèmes multi-agents»**, Mémoire Présenté pour l'obtention du diplôme De Magister en informatique, Université mohamed kheider Biskra, 2015 , pp 63-64. .

[33] : Abdelouahab BELAZOUI, **«Modélisation d'un comportement coopératif d'un agent»**, thèse Pour obtenir le titre de Docteur en Sciences Spécialité : Informatique, Université mohame kheider Biskra , 2016 , pp.93-95.

[34] : SAOULI Hamza, **«Découverte de services web via le Cloud computing à base d'agents mobiles»**, thèse pour l'obtention de grade de docteur 3eme cycle en informatique, Université mohamed kheider Biskra, 2015 , pp 120-121 .

[35] : https://fr.wikipedia.org/wiki/WinDev
[36] : https://www.emse.fr
[37] : https://doc.pcsoft.fr

[38] : BOULAHIA Nadia, **«Conception et réalisation d'une application à base des services web d'une agence de voyage en ligne»**, Mémoire de fin d'étude en vue de l'obtention du diplôme de master professionnel en informatique, Université Abderrahmane Mira de Béjaia, 2017, p 57 .

~Remerciements ~

A *louange appartient à* الله *je le loue et je le remercie pour m'avoir donné la force, la volonté et le Courage pour terminer ce modeste travail ;*

J*e tiens à remercier et à exprimer ma profonde gratitude à mon encadreur* ***M^{me} SAHLI Sihem*** *pour m'avoir fait confiance et accepter de m'encadrer au long de ce travail, pour sa disponibilité toujours en j'offrant ses conseils, pour son écoute et ses encouragements qu'il m'afflués.*

J*e remercie également les membres du jury qui ont accepté de juger mon travail.*

M*on infinie reconnaissance s'adresse à ma familles qui ont su m'apporter, sans relâche leur soutien durant toutes ces longue années d'études.*

E*nfin je remercie tous ceux qui ont contribués de près ou de loin à ce que ce modeste travail puisse voir le jour.*

Fairouz. L

الملخص

يُقصد بالتكيف مع معرف المستخدم ، من ناحية ، تسهيل التعبير عن احتياجات المستخدم والبحث عن معلومات حول موضوع ما عن طريق تجاهل المعلومات الغير مهمة وبالتالي تقليل مساحة البحث الخاصة بالمستخدم بشكل كبير .وثانيا ، لجعل هذه المعلومات المحددة مفهومة وقابلة للاستغلال للمستخدم .ومع ذلك ، فإن أهمية المعلومات ليست تدبيرا موضوعيا ، يمكن تعميمها على جميع المستخدمين .حيث يتم تعريفها من خلال مجموعة من المعايير والتفضيلات القابلة للتخصيص الخاصة بكل مستخدم أو مجتمع مستخدمين مقدم بواسطة معرف مستخدم محدد.

في هذا العمل ، إهتممنا بتكييف نتائج خدمات الويب وفقًا لتغيير معرفات المستخدمين حيث حاولنا اقتراح نهج قائم على الوكيل لتكييف خدمة الويب .من أجل التحقق من صحة هذا النهج قمنا بتنفيذ دراسة حالة باستخدام برنامج éclipse ومنصة jade .

الكلمات الأساسية: خدمة الويب، تكييف الخدمات، معرف المستخدم، الوكيل الاصطناعي

Abstract

Adapting to the user profile is intended, on the one hand, to facilitate the expression of the user's needs and to search for information on a subject by discarding irrelevant information and thus considerably reducing the search space of the user. And, secondly, to make this selected information intelligible and exploitable to the user. The relevance of the information is not generalizable to all users. It is defined by a set of criteria and customizable preferences specific to each user or community of users presented by a specific profile.

In this work, we have been interested in adapting the result of web services according to changing user profiles where we have tried to propose an agent-based approach for web service adaptation. In order to validate this approach we have implemented a case study using the eclipse IDE and the jade platform.

*Keyword***:** Web service, adaptation of services, profile of user, artificial agent.

Résumé

S’adapter au profil utilisateur a pour objectif, d’une part, de faciliter l’expression des besoin d’utilisateur et de rechercher des informations sur un sujet en écartant l’information non-pertinente et donc de réduire considérablement l’espace de recherche et, d’autre part, de rendre cette information sélectionnée intelligible à l’usager et exploitable. La pertinence de l’information n’est pas généralisable à tous les utilisateurs. Elle se définit par un ensemble de critères et de préférences personnalisables spécifiques à chaque utilisateur ou communauté d’utilisateurs présenté par un profil spécifique.

Dans ce travail, nous nous sommes intéressés à adapter le résultat des services web au changement des profils des utilisateurs où nous avons essayé de proposer une approche basée agent pour l’adaptation de service web. Afin de valider cette approche nous avons implémenté une étude de cas en utilisant l’IDE éclipse et la plateforme jade.

Mot clé: Service web, adaptation des services, profile utilisateur, agent artificiel.

INTRODUCTION GENERALE

CONCLUSION GENERALE ET PERSPECTIVES

CHAPITRE 1

L'ARCHITECTURE ORIENTÉE SERVICES ET LES SERVICES WEB

CHAPITRE 2

L'ADAPTATION DES SERVICE WEB

CHAPITRE 3

SYSTÈME MULTI AGENTS

CHAPITRE 4

CONCEPTION

CHAPITRE 5

IMPLÉMENTATION

RÉFÉRENCES BIBLIOGRAPHIQUE

TABLE DES MATIERES

CHAPITRE 3 : SYSTÈME MULTI AGENTS