République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A. Mira- Béjaia

Faculté des Sciences Exactes

Département d'Informatique

Mémoire de fin de cycle

En vue de l'obtention du diplôme de Master en Informatique

Option : Réseaux et Systèmes Distribués

Thème

Sélection et composition de services sensible au contexte avec QoS globale dans les environnements ubiquitaires

Mémoire soutenu le 25/06/2014 par :

M^{elle} AREZKI Yasmine.

M^{elle} BENSEGHIR Lynda.

Devant le jury composé de :

Présidente	**M^{me} GHANEM Souhila**	**M.A.B**	**Université de Béjaia.**
Rapporteur	**M^{me} KOUICEM Amel**	**M.A.B**	**Université de Béjaia.**
Co-Rapporteur	**M^{r} KHANOUCHE M.Essaid**	**M.A.A**	**Université de Béjaia.**
Examinatrice	**M^{me} BOUTRID Samia**	**M.A.A**	**Université de Béjaia.**
Examinatrice	**M^{me} DJERROUD Souhila**	**M.A.A**	**Université de Béjaia.**

Promotion 2013/2014

Remerciements

Nos vifs remerciements vont d'emblée à Dieu tout puissant qui nous a doté d'une grande volonté et d'un savoir adéquat pour mener à bien ce modeste travail.

Nos remerciements sont adressés également à nos chers parents pour tous les sacrifices consentis à notre égard et leur énorme

soutien.

A tous nos proches.

A nos encadreurs, ***M^{me} KOUICEM Amel*** *et* ***M^{r} KHANOUCHE M.E*** *qui nous ont inculqués une grande confiance et nous ont orienté dans le bon sens quant a l'élaboration de ce projet.*

Aux membres de la commission pour avoir accepté de juger notre modeste travail.

A tous nos enseignants et les membres du département informatique de l'université ABDERAHMENE MIRA.

Et enfin à tous ceux qui ont participé de près ou de loin à l'accomplissement de ce projet.

Dédicaces

Je dédie ce modeste travail :

A *ma chère maman, sans tes sacrifices, ta tendresse et ton affection je ne pourrais arriver jusqu'au bout. .*

A *mon cher papa, sans ton soutien je ne serais rien devenue. J'essaierai toujours d'être à la hauteur de ta confiance.*

A *ma chère et unique sœur* ***Sabrina*** *que j'aime beaucoup.*

A *mon beau frère Hocine.*

A *mes grands parents à qui je souhaite une longue vie.*

A *mes oncles et tantes.*

A *mes cousins et cousines.*

A *mon binôme* **LYNDA** *avec qui j'ai partagé des moments de joie.*

A *tous mes amis.*

A *tous tous ceux que j'aime.*

YASMINE

Dédicaces

A *la mémoire de mes grands-parents et mon cher oncle.*

A *mes très chers parents qui ont toujours été là pour moi, et qui m'ont donné un magnifique modèle de labeur et de persévérance. J'espère qu'ils trouveront dans ce travail toute ma reconnaissance et tout mon amour.*

A *mon unique frère adoré* ***Mouhamed el-amine****.*

A *mes chères sœurs* ***Nanou****,* ***Nissa*** *et* ***Ibtissem****.*

A *mes tantes, à mes oncles et leurs enfants.*

A *toute ma famille de près ou de loin.*

A *tous les étudiants de master 2 informatique (promotion 2014).*

A YASMINE *avec qui j'ai partagé des moments de joie.*

A *tous mes amis.*

Je dédié ce mémoire.

LYNDA

Table des matières

Liste des tableaux

Table des figures

Liste des abréviations

AGoSC	**A**bstract **G**raph of **S**ervice **C**omposition
API	**A**pplication **P**rogramming **I**nterface
AS	**A**bstract **S**ervice
ASD	**A**bstract **S**ervice **D**irectory
BPEL	**B**usiness **P**rocess **E**xecution **L**anguage
CS	**C**oncret **S**ervice
CSD	**C**oncret **S**ervice **D**irectory
DQP	**D**ynamic **Q**uality **P**arameters
DSSC	**D**ynamic **S**ervice **S**election **C**omposition
GPRS	**G**eneral **P**acket **R**adio **S**ervice
HTN	**H**ierarchical **T**ask **N**etwork
IA	**I**ntelligence **A**rtificielle
IHM	**I**nterface **H**omme **M**achine
MDP	**M**alkov **D**ecision **P**rocesses
MIP	**M**ixed **I**nteger **P**rogramming
PCM	**P**rocessus oriented **C**ontext **M**anager
PDA	**P**ersonal **D**igital Assistant
QdS	**Q**alité de **S**ervice
QoC	**Q**uality of **C**ontext
QoS	**Q**uality of **S**ervice
RFID	**R**adio **F**requency **ID**entification
SMA	**S**ystèmes **M**ulti **A**gents
SOA	**S**ervice **O**riented **A**rchitecture

SPSE	**S**ervice **P**roviders **S**earch **E**ngine
SQP	**S**tatic **Q**uality **P**arameters
SubGoSC	**S**ubgraph of **S**ervice **C**omposition
UDDI	**U**niversal **D**escription **D**iscovery and **I**ntegration
ULE	**U**biquitous **L**earning **E**nvironnement

Introduction Générale

La diversité des outils numériques utilisés dans nos activités quotidiennes, leurs nombres, leurs capacités de traitement et de communication sont en croissance perpétuelle. Ces dispositifs (téléphones à écran tactile, tablettes numériques, etc.) sont de plus en plus présents dans la vie de tous les jours. Ils constituent des outils d'assistance, de service, de communication et d'information, ils sont devenus à présent incontournables pour beaucoup d'entre nous. Ces évolutions technologiques permettent de concevoir des systèmes intelligents offrant une multitude de fonctions accessibles, à n'importe quel moment, et à n'importe quel endroit et selon une multitude de modes d'interactions et de média. On parle alors de systèmes ubiquitaires ou omniprésents, ou de systèmes intelligents ambiants. L'objectif est de créer des environnements intelligents permettant d'améliorer l'environnement de vie des usagers. Ainsi dans le cadre des applications de maintien à domicile des personnes âgées ou dépendantes afin d'améliorer leurs qualité de vie.

Une caractéristique importante de ces systèmes, qui leurs permet d'interagir de manière transparente avec les usagers et de les assister dans leurs tâches de tous les jours, est la sensibilité au contexte. Le contexte dans un système ubiquitaire regroupe des informations et des évènements qui ont lieu dans l'environnement des entités du système informatique. Ces entités sont : les utilisateurs, les dispositifs, les logiciels, etc. Les informations du contexte sont utilisées par le système informatique pour fournir du soutien et pour bien réagir aux demandes des utilisateurs.

La composition dynamique de services est la constitution automatisée d'un service composite à partir d'un ensemble services atomiques existant. Les services composites résultants fournissent de nouvelles fonctionnalités qui ne sont pas directement disponibles dans les services composants élémentaires. Ainsi, la sélection de services correspond à la recherche d'une combinaison de services élémentaires les mieux

adaptés aux exigences et préférences de l'utilisateur en tenant en compte de la QoS qui est un facteur important dans la composition de services.

La problématique traitée dans ce travail, est la sélection et la composition dynamique de services sensible au contexte et avec qualité de services (QoS). Cette composition, nécessite la prise en compte de plusieurs facteurs, à savoir la QoS, les préférences de l'utilisateur et le dynamisme des environnements ainsi que la prise en considération des informations contextuelles.

Ce mémoire est structuré en quatre chapitres, le premier chapitre constitue le contexte général de ce travail. Il s'agit de présenter les systèmes ubiquitaires, leur principe et leurs caractéristiques et l'importance de la sensibilité au contexte dans ces environnements.

Dans le deuxième chapitre, nous donnons tout d'abord un ensemble de définitions relatives à la composition de services dans les environnements ubiquitaires, puis nous nous focalisons sur la problématique de la sélection et la composition de services. Nous passons également en revue quelques approches de composition proposées dans la littérature.

Le troisième chapitre détaille notre proposition à la problématique cités précédemment tout en prenant en compte les préférences de l'utilisateur et les contraintes globales en termes de qualité de services, et la sensibilité au contexte lors de la sélection.

Le quatrième chapitre, quant à lui, décrit l'implémentation et la validation de l'approche proposée. Dans ce chapitre, les performances de l'approche proposée sont également évaluées et analysées. Nous finisons ce chapitre par un scénario illustrant notre contribution.

Enfin, nous concluons ce travail en dégageant quelques perspectives de recherche.

Chapitre 1

Présentation des systèmes ubiquitaires

1.1 Introduction

Grâce à la mise en réseau d'un nombre croissant de dispositifs informatiques et l'émergence de technologies sans fil, l'informatique ubiquitaire (ubiquitos computing) devient aujourd'hui une réalité.

Le dictionnaire définit le terme ubiquitaire comme "quelque chose qui est ou semble être partout à la fois"[1]. L'informatique ubiquitaire consiste à offrir aux utilisateurs un accès aux services de leurs environnements immédiat quelque soit l'hétérogénéité matérielle et logicielle des dispositifs informatiques sous-jacents [2] . L'objectif est de permettre à ces derniers de consulter des données n'importe quand, n'importe où et à partir de n'importe quel périphérique.

Dans ce qui suit, nous allons présenter en premier lieu l'informatique ubiquitaire : principe, environnement, les défis et quelques domaines d'application. En deuxième lieu le contexte et la sensibilité au contexte dans ces environnements.

1.2 Les systèmes ubiquitaires

1.2.1 Principe de l'informatique ubiquitaire

L'informatique ubiquitaire est aussi appelée informatique diffuse, informatique pervasive, intelligence ambiante et aussi informatique omniprésente. Elle peut être vue comme un domaine particulier de l'in-

formatique résultante de la convergence des travaux existants dans les domaines de l'informatique mobile et des systèmes distribués [3].

L'informatique pervasive est une vision d'actualité, dans laquelle un nombre croissant d'appareils inclus dans divers objets physiques participent à un réseau d'information global. L'un des aspects les plus novateurs de l'informatique ubiquitaire est qu'elle essaye d'interagir avec l'utilisateur de la façon la plus naturelle possible.

Ses origines remontent à 1991, lorsque Mark Weiser présentait sa vision futuriste de l'informatique du 21eme siècle en établissant les fondements de l'informatique pervasive, pour que les dispositifs doivent pouvoir s'adapter naturellement à la situation de l'utilisateur. Les fonctionnalités offertes par les dispositifs informatiques peuvent être abstraites sous forme de services [4].L'objectif est de permettre aux utilisateurs d'accéder aux différentes fonctionnalités offertes par les divers dispositifs informatiques présents dans leur environnement immédiat, à partir de n'importe quel terminal comme par exemple leur téléphone portable, leur PDA (Personal Digital Assistant).

L'informatique ubiquitaire peut être vue comme l'opposé de la réalité virtuelle. La réalité virtuelle met une personne à l'intérieur d'un monde créé par l'ordinateur. L'informatique ubiquitaire a pour but de permettre à l'ordinateur de vivre dans le monde des hommes et de s'y intégrer .

FIGURE 1.1 – Quelques équipements de l'informatique ubiquitaire.

1.2.2 Scénario illustratif

On dispose d'une maison intelligente équipée de tous les outils nécessaires pour aider des personnes dépendantes, tels que : les capteurs, des caméras de surveillance, des robots, des systèmes de reconnaissance vocale, des outils média, ...etc.

Notre scénario consiste à rappeler les horaires de traitement pour une personne atteinte de la maladie d'Alzheimer à partir de la prescription du médecin et du contexte courant.

Le système localise l'endroit ou se trouve la personne en utilisant le service de localisation et l'informe du moment de la prise de ses médicaments soit, par un message vocal ou par message textuel , cela dépend du média proche de cette personne.

Une fois que la personne est informée, elle prend place et attend l'arrivé de son médicament.

Par la suite, le robot sera informé de l'endroit de la personne et celui des médicaments. A partir de ces informations, le robot récupère les médicaments et les transporte à la personne.

1.2.3 Les défis de l'informatique ubiquitaire

La multiplication des terminaux mobiles et la généralisation des réseaux sans fil dans l'informatique ubiquitaire soulèvent de nombreux problèmes :

1.1.3.1 Hétérogénéité

Les réseaux ubiquitaires sont composés de dispositifs généralement hétérogènes pouvant être mobiles ou non. Cette hétérogénéité a des conséquences sur les réseaux ubiquitaires : à titre d'exemple, on ne peut pas demander aux petits processeurs placés dans un MP3 de réaliser les mêmes opérations qu'un ordinateur de bureau. D'autre dispositifs tels que les assistants numériques personnels présentent de meilleures capacités de traitement mais soufrent d'un stockage limité. [5].

1.1.3.2 Limitation des réseaux

La connexion sans fil utilisée dans les équipements mobiles est souvent limitée en bande passante et instable par rapport à la connexion filaire. A présent les technologies standards sans fil incluent le Bluetooth [6] , wifi (standard 802.11), GPRS (General Packet Radio Service), et d'autres en cours de développement. La vitesse lente de ces technologies par rapport au réseau filaire a toujours été une barrière pour exécuter de grande application et déplacer des données volumineuses sur le réseau mobile.

1.1.3.3 Dynamisme

L'une des caractéristiques des environnements ubiquitaires est la forte mobilité des terminaux ainsi que les utilisateurs [7]. En effet, les utilisateurs passent de réseau en réseau ; ces déplacements provoquent des changements de topologie. Le gestionnaire de contexte doit assurer une continuité de service malgré les déplacements des terminaux avec changement de réseau et déconnexion.

1.1.3.4 Découverte de services

Il faut permettre à un utilisateur de savoir que des services sont disponibles en un lieu précis, en fonction de la géo-localisation. De plus, le code des services disponibles à déployer sur son terminal est différent selon le terminal utilisé par exemple un PDA noir et blanc ou un ordinateur portable couleur. Pour faire ce choix, le système doit avoir des informations sur les caractéristiques du terminal afin de présenter à l'utilisateur un sous-ensemble de services qui potentiellement peuvent s'exécuter sur son terminal [8].

1.1.3.5 Déploiement de services

Dès qu'un utilisateur est intéressé par un service, l'infrastructure doit être capable de déployer le code de ce service sur son terminal. Plus précisément, la partie cliente du logiciel implantant le service, par exemple : l'IHM affichant la liste des trains au départ, est téléchargée puis instanciée sur le terminal à la demande de l'utilisateur. La partie fixe du service, par exemple : le système d'information de la gare et le code du traitement pour l'extraction et l'envoi des données, est supposée avoir été déployée préalablement [8].

1.2.4 Domaines d'application

Il existe de nombreuses applications de l'informatique pervasive qui touche quasiment tous les domaines, d'aprés notre thématique nous allons citer les domaines suivants :

1.1.4.1 Domotique

Une maison intelligente ou smart home est une résidence équipée de technologie d'informatique ambiante, qui anticipe et répond aux besoins de ses occupants en essayant de gérer de manière optimale leurs conforts et leurs sécurités par action sur la maison, et en mettant en œuvre des connexions avec le monde extérieur [9] . La maison intelligente est donc une branche importante de la domotique.

1.1.4.2 Domaine médicale

La technologie du soin portable dans l'environnement ubiquitaire fournit un service médical personnalisé n'importe quand et n'importe où afin qu'une meilleure qualité de vie puisse être atteinte. Dans[10], l'auteur propose un modèle conceptuel d'un système, qui joue un rôle important dans la prévention des maladies, précocité du diagnostic ou l'efficacité du traitement ce qui peut apporter des avantages considérables à la société. Le groupe K.Ouchi et al, ont proposé un modèle conceptuel pour la construction d'un système de soins dans une voiture mobile (smart hospital) qui est LIFEMINDER. Ce dernier est un modèle médical pour la surveillance des changements d'état d'un patient ; il est d'un modèle capteur (WristWatch-sharped wearable sensor) et d'un PDA. Le capteur sert à mesurer les trois axes physiologique tel que : l'accélération du sang, le taux de la pulsation du cœur et la température du corps. Le PDA sert à reconnaitre les activités du patient comme la marche et la nutrition via un réseau Bluetooth. La connaissance de ces activités réduit les problèmes physiques et psychiques de patient.

1.1.4.3 Domaine publique

L'exemple le plus simple est le domaine des transports. Dans ce cadre plusieurs projets de travail ont été réalisés comme le projet Assistant Paris Voyage Metro, ce projet est une application sensible au contexte mobile développé conjointement par l'université Ryerson et Appear Networks [11].

Le logiciel permet aux passagers du métro de trouver leur chemin en fournissant des cartes et recommander des itinéraires de rechange lorsque les aspects du système sont en baisse. Il conserve également les utilisateurs informés sur les points d'intérêt à travers la ville. Les passagers s'abonnent à une liste des événements. Un événement peut être un concert (définie soit par un groupe de musique spécifique ou par un genre de musique). Il peut également être un restaurant qui est un endroit particulier ou un type de restaurant (restauration rapide). Avec la liste des événements mise en place, l'Assistant Paris Voyage Métro avertit dès qu'ils sont à proximité.

1.1.4.4 U-Learning

Un environnement d'apprentissage ubiquitaire (en anglais ULE Ubiquitous Learning Environment) est un paramètre d'apprentissage. Des microprocesseurs sont intégrés dans les dispositifs où l'utilisation des technologies sans fis et mobiles facilitent l'accès aux fonctionnalités éducatives. L'ULE peut fournir des accessoires et des outils pour encourager la participation des élèves [12] . Il permet l'intégration et la coordination des institutions pédagogiques (école, université, laboratoire de recherche etc.), les lieux du

travail, les communautés et les familles des élèves. Les élèves, éducateurs et les parents communiquent, collaborent et se coordonnent pour avoir une bonne qualité pédagogique avec des appareils qui peuvent être mobile ou immobile tels que les RFID, les PDA, les téléphones mobiles et les PC portables [13].

1.3 L'architecture Orientée Services (SOA)

Pour faciliter le développement d'applications réparties et pour assurer une interopérabilité entre les applications pour un tel environnement, le paradigme le plus approprié est l'architecture orienté service SOA (service oriented architecture) [14].

Une architecture orientée services (notée SOA) est une architecture logicielle s'appuyant sur un ensemble de service simples [15] . Elle fournit un ensemble de méthodes pour le développement et l'intégration de systèmes dont les fonctionnalités sont développées sous forme de services interopérables et indépendants. Une infrastructure SOA vise à permettre l'échange d'informations entre les applications [5]. SOA se base sur des concepts plus anciens tel que l'informatique distribuée.

Ce style d'architecture convient le mieux à l'environnement de l'informatique ubiquitaire, qui permet une utilisation homogène de composants logiciels hétérogènes présents dans l'environnement.

1.4 Contexte et sensibilité au contexte

1.4.1 Définition du contexte

Le mot contexte provient du latin contexte qui signifie : tisser ensemble. Le dictionnaire Petit Larousse le définit comme : Ensemble des circonstances, situation globale ou se situe un événement.

La notion du contexte désigne en général l'ensemble des informations qui entourent une activité et des informations supplémentaires sur son environnement. Dans leurs interactions, les humains utilisent et tiennent compte de leur contexte d'une manière implicite pour mieux se comprendre et changer leurs comportements [16].

Dans le domaine de l'informatique ubiquitaire, la notion du contexte prend beaucoup d'ampleur.

Plusieurs définitions ont été proposées, qui sont si spécifiques et qui ont été énoncées en fonction des besoins propres des domaines d'application. On peut conclure que le contexte est toutes les informations sur l'ensemble des états de l'application et l'ensemble d'informations sur ses utilisateurs [17].

1.4.2 La sensibilité au contexte

Un système informatique sensible au contexte est un système ayant la possibilité de percevoir la situation de l'utilisateur dans son environnement et d'adapter par conséquent tout le comportement du système à la situation en question : ses services, ses données et son interface utilisateur [4].

Dans différent contextes les utilisateurs accèdent aux mêmes données et aux mêmes services mais reçoivent des réponses qui peuvent être différentes ou présentées différemment ou à des niveaux de détails différents [18]. Par exemple, un docteur consulte le dossier médical d'un patient à l'hôpital à l'aide d'un ordinateur de bureau. Dans un autre contexte, il consulte le même dossier mais chez le patient à l'aide d'un ordinateur de poche. Dans les deux cas le résultat obtenu est différent. D'une façon générale, le contexte décrit la situation de l'utilisateur en termes de localisation, de temps, d'environnement, de terminal utilisé, de profil utilisateur, etc.

1.5 La qualité de service (QoS)

Un service possède généralement un ensemble de paramètres de qualité de service, même si beaucoup d'entre eux sont de nature dynamique, c'est à dire, lié à son environnement d'exécution.

Politiques de qualité de service sont les règles relatives aux paramètres de qualité de service, tel que le temps de réponse. La politique de qualité de service comprend à la fois des propriétés fonctionnelles et non fonctionnelles[15].

Les propriétés fonctionnelles reflètent le fonctionnement du service, tandis que les propriétés non fonctionnelles sont l'aptitude d'un service à répondre adéquatement aux exigences de l'utilisateur, dans le but de le satisfaire. Ces exigences peuvent être liées à plusieurs propriétés non-fonctionnelles à savoir : la disponibilité, la fiabilité, le temps de réponse, la sécurité. Etc.

1.6 Conclusion

Dans ce chapitre, nous avons passé en revue des différents aspects des environnements ubiquitaires. Nous avons présenté les différentes caractéristiques de ces environnements, et mis en évidence les besoins des applications ubiquitaires, notamment les problèmes d'hétérogénéité, la prise en compte du contexte et la dynamicité à travers le scénario présenté.

Plusieurs solutions ont été proposées dans la littérature. Le chapitre suivant sera consacré à la description de quelques solutions du problème de sélection et de composition de services en prenant en considération certaines contraintes.

Chapitre 2

Etat de l'art sur la composition de services

2.1 Introduction

L'environnement de l'informatique ubiquitaire est constitué de plusieurs dispositifs intelligents qui offrent des services aux utilisateurs. La découverte de ces services est importante dans cet environnement, où les ressources et les services disponibles changent selon la mobilité de ces derniers. En effet si l'utilisateur demande une requête qu'aucun service n'est apte à l'a réalisée. Dans ce cas, il est nécessaire de combiner les services atomiques existants pour répondre à sa requête tout en prenant compte son contexte tels que le temps, la localisation et ses préférences.

La composition de service consiste à créer des services complexes en combinant des services atomiques existants prenant en compte les besoins de l'utilisateur et le contexte de l'environnement. En effet, le compositeur de service définit une action appropriée et planifier pour effectuer une tache de composition et déterminer en suite le service le plus approprié pour chaque action du plan avec des qualités de services meilleurs.

Ce chapitre représente l'état de l'art de la composition de services. Nous donnons tout d'abord quelques définitions qui concernent la composition de services et les différentes méthodes de composition. Puis nous allons présenter et critiquer des propositions existantes pour la composition de services avec QoS en environnements ubiquitaires.

2.2 Composition de services

2.2.1 Définitions

2.1.1.1 Définition d'un service

Un service est une entité logicielle qui permet d'exposer une ou plusieurs fonctionnalités définies par une interface. Cette interface comporte les propriétés fonctionnelles et éventuellement des propriétés non-fonctionnelles d'un service permettant d'identifier le service qui convient le mieux aux besoins des utilisateurs [19].

2.1.1.2 Les propriétés d'un service

L'identité d'un service est définie par ses propriétés fonctionnelles et non fonctionnelles. En effet, les services se distinguent par les fonctionnalités qu'ils peuvent fournir. Les propriétés Non fonctionnelles sont la base de définition de la qualité de service [19].

- **Les propriétés fonctionnelles :**
  Elles sont décrites dans la description de service en termes d'opérations, et reflètent le fonctionnement du service.
- **La qualité de service :**
  La Qualité de service (Quality of Service ou QoS) est l'aptitude d'un service à répondre adéquatement et dans de bonnes conditions aux exigences de l'utilisateur, dans le but de le satisfaire. Ces exigences peuvent être liées à plusieurs propriétés non-fonctionnelles d'un service à savoir : la disponibilité, la fiabilité, le temps de réponse, la sécurité. Etc.

2.1.1.3 Définition de la composition de services

La composition de services consiste à combiner les fonctionnalités de plusieurs services dans le but de répondre à des demandes complexes qu'un seul service ne pourrait pas satisfaire. La composition de services vise à faire intéropérer, interagir et coordonner plusieurs services pour la réalisation d'un but donné [20].

La figure suivante illustre le principe de la composition de services à partir d'un ensemble de services disponibles dans un registre (annuaire).

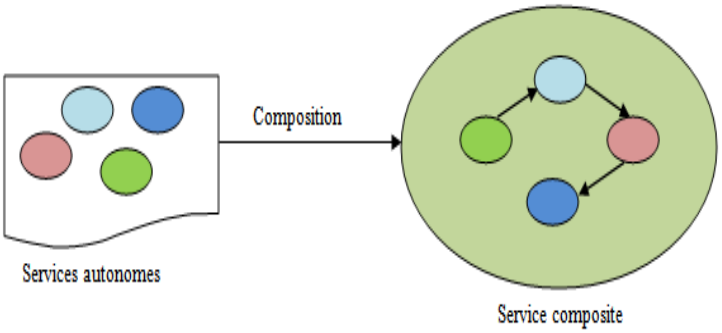

FIGURE 2.1 – Principe de la composition de services.

2.2.2 Méthodes de composition de services

Les solutions proposées pour la composition de services peuvent être classifiées selon deux axes :

- Les approches de composition statiques.
- Les approches de composition dynamiques.

Le schéma ci-dessous illustre les differentes méthodes de composition de services :

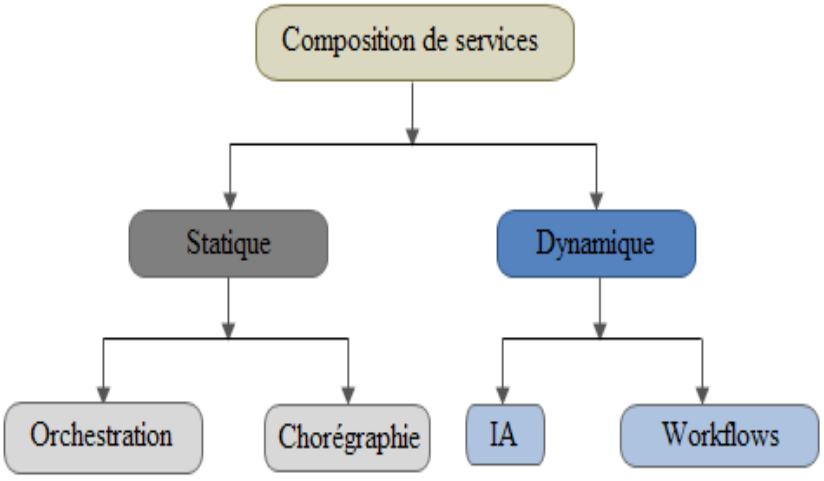

FIGURE 2.2 – Méthodes de composition de services.

2.1.2.1 Approches de composition statique

- **Orchestration**

  L'orchestration décrit un ensemble d'actions dans lesquelles un service donné peut ou doit s'engager. Elle offre une vision centralisée de la logique de coordination d'une composition de services. Un coordinateur central - appelée moteur d'exécution- prend le contrôle de tous les services intervenant dans une composition de services. Il gère l'invocation de ces différents services selon la logique définie par le workflow [21].

  La figure suivante montre le workflow dans l'orchestration des services.

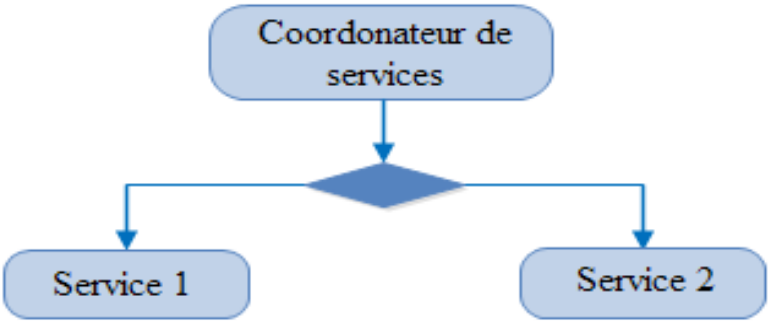

FIGURE 2.3 – L'orchestration de services.

- **Chorégraphie**

  Contrairement à l'orchestration, la chorégraphie n'a pas un coordinateur central. Chaque service mêlé dans la chorégraphie connaît exactement quand ses opérations doivent être exécutées et avec qui l'interaction doit avoir lieu [22].

  La collaboration dans la chorégraphie des services peut être représentée de la manière suivante :

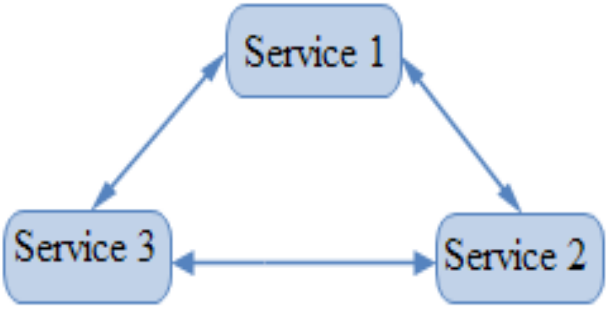

FIGURE 2.4 – La chorégraphie de services.

2.1.2.2 Approches de composition dynamique

Les différentes approches existantes pour la composition dynamique de services peuvent être regroupées en deux courants : les approches basées sur les workflow et les approches basées sur les techniques de l'intelligence artificielle.

1. **Les approches orientées workflow**

   Se basent sur le fait qu'un service composite est définit par un ensemble de services atomiques et par la façon dont ils communiquent entre eux.

   Ce courant propose donc d'adapter les méthodes d'orchestration et de chorégraphie afin de les rendre dynamiques.

2. **Les approches orientées intelligence artificielle**

   - **Calcul situationnel**

     Dans ce type d'approche, Le problème de la composition est abordé de la façon suivante : la requête de l'utilisateur et les contraintes des services sont représentées en terme de prédicats du premier ordre dans le langage de calcul situationnel. Les services sont transformés en actions (primitives ou complexes) dans le même langage. Puis, à l'aide de règles de déduction et de contraintes, des modèles sont ainsi générés et sont instanciés à l'exécution à partir des préférences utilisateur [23].

- **Preuve de théorèmes**

  dans ce type d'approche, les services disponibles et les requêtes des utilisateurs sont traduites dans un langage du premier ordre. Puis des preuves sont produites à partir d'un prouveur de théorèmes [23].

- **Composition avec SMA**

  La composition de services peut être implémentée aussi en utilisant des SMA. Dans cette approche, chaque agent présente un service et sert à satisfaire une partie de la requête de l'utilisateur en utilisant ses propres capacités [22] .

- **Composition par planification**

  La planification est l'un des domaines de l'Intelligence Artificielle (IA) qui permet de choisir et d'organiser des actions, en fonction d'un but donné. Le mot planification est également utilisé dans plusieurs autres domaines, tels que l'économie, la politique, l'architecture et encore dans la vie de tous les jours.

  La planification en IA peut aussi être conceptualisée comme un choix et une organisation d'actions pour changer l'état d'un système. Par extension, elle produit un plan qui est présenté sous la forme d'une collection de descriptions d'opérateurs. Le planificateur ou générateur de plans est le système qui produit un plan. L'exécution est la réalisation des actions présentes dans le plan. L'exécution d'un plan modifie les propriétés du monde en le faisant évoluer de l'état initial jusqu'au but désiré [24].

2.3 Approches de composition de services

2.3.1 La combinaison de l'optimisation globale et la sélection locale pour une composition de services avec QoS

2.2.1.1 Description

Dans cette solution [25], les auteurs ont abordé le problème de composition de services Web par une solution qui combine l'optimisation globale et les techniques de sélection locale en exploitant les avantages des deux.

- **Les contraintes locales :** représentent toutes contraintes concernant un service composant sé-

lectionné pour le service composite, par exemple le temps de réponse, la fiabilité... etc.

- **Les contraintes globales :** représentent les contraintes concernant tout le service composite, par exemple le temps de réponse total du service composite.

Après avoir normalisées toutes les valeurs des paramètres de chaque service, la solution se déroule en deux phases :

Phase(1) : décomposition des contraintes globales de QoS

En démarrant de l'hypothèse que la satisfaction des contraintes locales de QoS garantit la satisfaction des contraintes globales, le problème de décomposition peut être modélisé sous forme d'un problème d'optimisation des contraintes de QoS dont l'objectif est de trouver un ensemble de contraintes locales pour chaque classe de service qui couvrent autant de services candidats que possibles , tandis que leur agrégation ne viole aucune des contraintes globales . En effet, deux étapes sont utilisées :

- **Détermination de niveau de QoS**

  Les niveaux de qualité sont initialisés pour chaque classe de services, en divisant les intervalles de valeurs de chaque attribut QoS_{qk} en un ensemble de valeurs de qualité discrètes d. Par la suite, la valeur de chaque niveau de qualité P^z_{jk} qui est entre 0 et 1 sera déterminée en appliquant la formule (2.1), après avoir calculé $\mathrm{h}(q^z_{jk})$ qui représente le nombre de services qui seraient admissibles si ce niveau a été utilisé comme une contrainte locale.

$$p^z_{ik} = \frac{h(q^z_{jk})}{l} * \frac{u(q^z_{jk})}{u_{max}} \tag{2.1}$$

  Où :

  l est le nombre total de services de la classe S_j, u_{max} est la valeur maximale de la fonction d'utilité, P^z_{jk} indique combien de services Web seraient admissibles si le z^{eme} niveau à été utilisé comme une contrainte locale pour le k^{eme} attribut de QoS dans la classe S_j.

- **formulation du modèle MIP**

  Le MIP est utilisé afin de trouver la meilleure décomposition des contraintes globales de QoS à des contraintes locales. Par conséquent ils utilisent une variable de décision binaire x^z_{jk} pour chaque niveau de qualité local q^z_{jk} tel que x^z_{jk}= 1 si q^z_{jk} est sélectionné comme une contrainte locale pour l'attribut q_k de QoS de la classe de services s_j, et x^z_{jk}= 0 sinon.

Phase(2) : la sélection locale

Après la phase précédente, chaque prestataire de service effectue une sélection locale et renvoie le meilleur service Web candidat pour la composition de services. Une liste de services qualifiés est créée et triée par leurs valeurs d'utilité qui seront calculées pour chaque service, en déterminant la différence entre les valeurs de QdS des services candidatas et les valeurs maximales de services de la même classe.

La figure suivante illustre les étapes de cette approche :

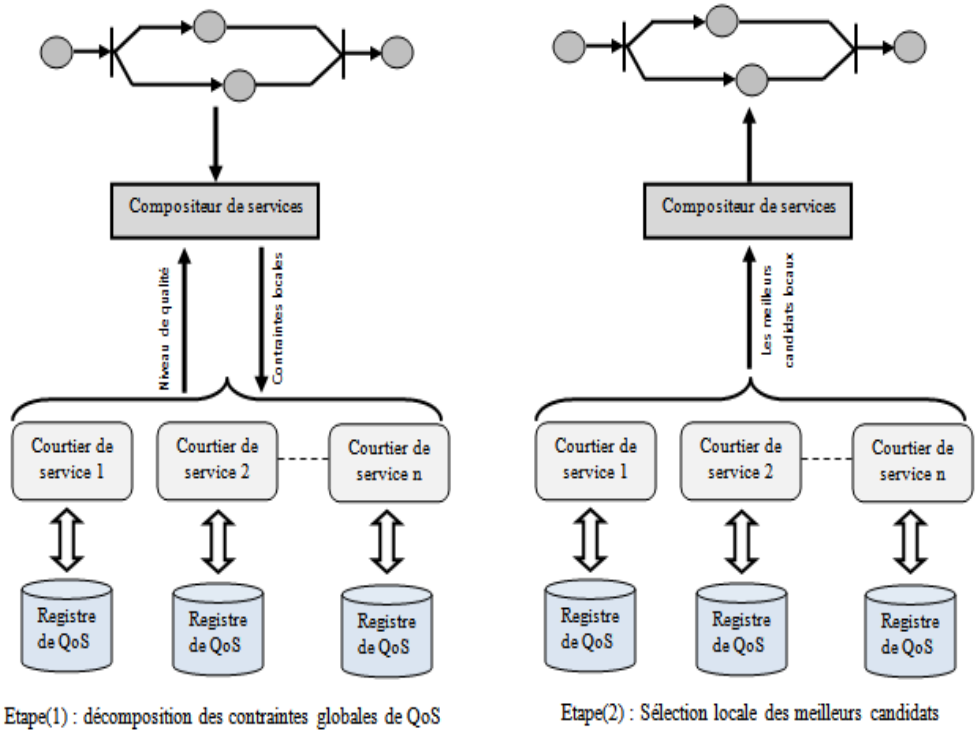

FIGURE 2.5 – Sélection de services distribuée sensible à la QoS.

2.2.1.2 Discussion

En combinant l'optimisation globale avec sélection locale cette approche est capable de résoudre efficacement le problème de sélection de manière distribuée, elle est particulièrement utile pour des applications avec des changements dynamiques et des exigences de temps réel et pour une composition efficace basée sur la qualité de service mais, les auteurs ont pris en considération que le modèle séquentiel et que les attributs négatifs de QoS. De même pour les niveaux de qualité de service qui sont déterminés aléatoirement ce qui reste un inconvénient pour cette solution.

2.3.2 Composition de services avec QoS dans les environnements dynamiques orientés services

2.2.2.1 Description

La question qui se pose est "comment faire face au dynamisme des environnements lors de la sélection, la composition et l'exécution des services avec QoS ?". Les auteurs de cette solution [26], ont présenté un algorithme de sélection efficace qui fournit le terrain adéquat pour la composition avec QoS dans les environnements dynamiques, en déterminant un ensemble de compositions quasi-optimales qui respectent les contraintes globales de QoS imposées par l'utilisateur, et optimisent la fonction d'utilité de la QoS. Pour cela, les auteurs ont classifié les attributs de QoS comme suit :

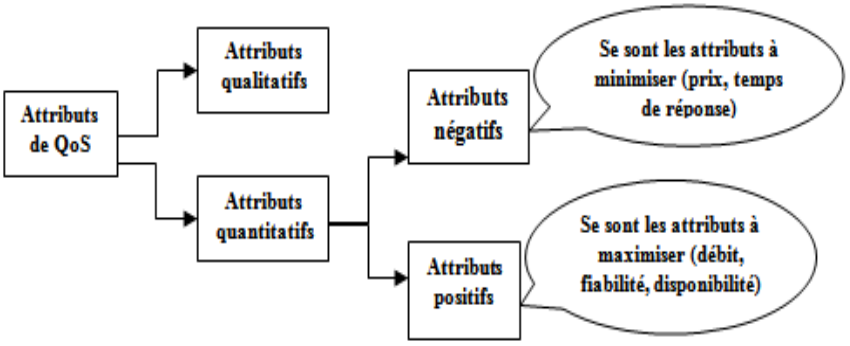

FIGURE 2.6 – Classification des attributs de qualité de services.

Le but de cet algorithme est de déterminer un ensemble de compositions quasi-optimales, les auteurs utilisent un modèle de composition de services qui englobe la plupart des structures spécifiées par les langages de composition (BPEL) [27, 28, 29, 30], à savoir le modèle séquentiel, parallèle, conditionnel et le modèle en boucle.

La QoS de chaque service abstrait est représentée en utilisant un vecteur $QoS_i = <q_{i,1},...,q_{i,n}>$, sachant que n est le nombre d'attributs de QoS requis et '$q_{i,j}$' représente la valeur de l'attribut de QoS 'j' du service 'i'. La composition de services est évaluée à base des vecteurs de QoS du service composite en tenant compte des modèles de composition, comme le montre le tableau suivant :

Attributs de QdS	Modèle de composition			
	SEQUENCE	ET	XOR	BOUCLE
Temps de réponse	$\sum_{i=1}^{n}$=Rt_i	Max(Rt_j)	Max(Rt_j)	Rt*K
Fiabilité	$\prod_{i=1}^{n}$=RE_i	$\prod_{i=1}^{n}$=RE_i	Min(RE_j)	RE_k
Disponibilité	$\prod_{i=1}^{n}$=AV_i	$\prod_{i=1}^{n}$=AV_i	Min(AV_j)	RE_k
Débit	Min(TH_j)	Min(TH_j)	Min(TH_j)	TH
Prix	$\sum_{i=1}^{n}$=P_i	$\sum_{i=1}^{n}$=P_i	Max(P_j)	P*K

TABLE 2.1 – Matrice de classification de services concrets.

L'algorithme de sélection de services

Lors d'une sélection de services, deux approches ont été définies : (i) la sélection locale qui consiste à sélectionner un meilleur service concret individuellement de chaque service abstrait ; (ii) la sélection globale permet de considérer toutes les compositions possibles des services afin de sélectionner la meilleure solution optimale, qui répond aux exigences de l'utilisateur en terme de QoS globale.

Dans cette solution, les auteurs ont proposé un algorithme heuristique qui utilise la sélection locale et la sélection globale. L'algorithme passe par les trois phases suivantes :

1. **Prétraitement**

   Vise à normaliser les valeurs des attributs de QoS en les transformant en des valeurs entre 0 et 1 en utilisant les formules suivantes :

   Attributs négatifs :

$$q'_{i,j} = \begin{cases} \frac{q_j^{max} - q_{i,j}}{q_j^{max} - q_j^{min}}, & si \quad q_j^{max} - q_j^{min} \neq 0 \\ 1, & sinon \end{cases} \tag{2.2}$$

   Attributs positifs :

$$q'_{i,j} = \begin{cases} \frac{q_{i,j} - q_j^{min}}{q_j^{max} - q_j^{min}}, & si \quad q_j^{max} - q_j^{min} \neq 0 \\ 1, & sinon \end{cases} \tag{2.3}$$

   Où :

   q'$_{i,j}$ est la valeur normalisée de la QoS associée à l'attribut j et le service candidat S_i, $q_{i,j}$ représente

la valeur courante, q_{max} et q_{min} sont les valeurs maximale et minimale respectivement des attribut de QoS parmi tous les services candidats.

2. **Classification locale**

   Elle a pour objectif de classifier tous les services candidats en fonction de leur niveau de QoS en utilisant la technique de clustering qui consiste à regrouper les services candidats qui possèdent à peu prées la même QoS dans un même cluster, comme illustré dans la figure suivante :

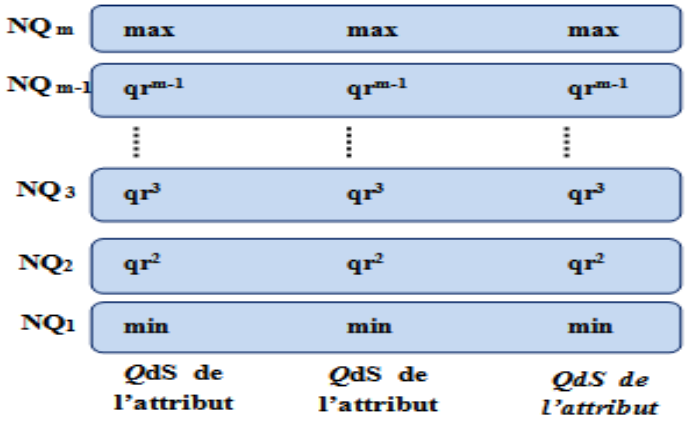

FIGURE 2.7 – Présentation des niveaux de QoS.

3. **Sélection globale**

   Dans cette phase, les services obtenus seront utilisés pour orienter le choix des compositions quasi-optimales qui maximises la fonction d'utilité et qui respecte les contraintes de QoS globales, en sélectionnant de chaque service abstrait un service concret.

2.2.2.2 Discussion

Dans cette proposition, les contraintes de QdS sont prises en considération durant la composition.

Cette approche a pris en considération la dynamicité des environnements durant la sélection, la composition et l'exécution des services en cas de défaillances ou d'échecs de l'un des services utilisé pour la composition. Tandis que la qualité des informations contextuelles qui n'est pas prise en considération dans cette solution.

2.3.3 Composition de services sensible au contexte dans les environnements ubiquitaires

2.2.3.1 Déscription

Dans cette solution [31], Les auteurs proposent une approche de conception en couches flexible et tolérante aux échecs pour la composition de services dans les environnements ubiquitaires. En effet, les auteurs supposent que :

Hypothèse1 :

Tous les services sont découverts et mis à disposition pour une éventuelle utilisation dans l'annuaire des services, ils considèrent deux types de services comme il est généralement considéré dans la littérature : Les services concrets sont enregistrés dans le répertoire des services concrets (CSD : Concret Service Directory). Ces services sont classés en fonction de leurs tâches dans les services abstraits publiés dans le répertoire des services abstraits (ASD : Abstrait Service Directory). Un service abstrait i noté SA_i. Il est décrit par un quadruplet $<SA_i^{in}, SA_i^{out}$, SA_{csi}, R$>$ tel que :

- SA_i^{in} : Ensemble des entrées de SA_i.
- SA_i^{out} : Ensemble des sorties de SA_i.
- R : Réputation du service abstrait.
- SA_{csi} : Ensemble des services concrets de SA_i tel que :

$$SA_{csi}=cs_{i,1},cs_{i,2},..,cs_{i,n} \text{ Et} \forall cs_{i,j},cs_{i,k} \epsilon SA_{csi} / (cs_{i,j}^{in}=cs_{i,k}^{in}=AS_i^{in}) \bigwedge (cs_{i,j}^{out}=cs_{i,k}^{out}=AS_i^{out}).$$

Hypothèse2 :

Le gestionnaire du contexte capture les attributs contextuels à partir du profil utilisateur et du contexte de son environnement et génère ensuite une requête T décrite sous la forme (T, T_{out}), où (T_{in}, T_{out}) représente les E/S de la requête T.

Hypothèse3 :

Chaque utilisateur a son propre profil qui est composé de : son statut, ses renseignements personnelles, ses préférences, les préférences de sécurité de l'utilisateur et le seuil QT minimal de la qualité globale d'un service que l'utilisateur peut accepter, le poids de chaque qualité de service en fonction de son importance pour l'utilisateur.

Les auteurs classent les paramètres de qualité en deux classes : les paramètres de qualité de service(QoS) et les besoins et préférences (qualité globale) exigé par l'utilisateur (QoE).

L'approche de composition développée dans cet article comprend trois couches :

Première couche : génération du plan abstrait

La génération d'un plan global consiste à la construction automatique d'un plan de composition abstrait, en utilisant une technique à base de règles. Le plan contient tous les services abstrait possible qui peuvent participer à la composition du service exprimé dans la requête T.

Initialement, le graphe G =(V, E) =(RP_0,). RP_0 étant les paramètres de sortie de la requête T. Pour chaque paramètre RP_i demandé, les services abstraits sont choisis à partir du répertoire des services abstraits (ASD). Ces services abstraits sélectionnés, appelés services abstrait candidats, ont au moins un paramètre de sortie appartenant à RP telle que :

- RP_i : est le i^{me} ensemble des paramètres requis, sachant que l'ensemble initial RP_0 représente les paramètres de sortie nécessaires de la requête T.
- Un arc de RP_i à RP_j noté (RP_i, RP_j, AS_i, R_i) représente le service abstrait qui fournit tous ou une partie des paramètres requis RP_i et qui requiert le paramètre RP_j Le service abstrait AS_i a une réputation RP_i.

Si toutes les données fournies par ces services abstraits candidats comprennent les paramètres RP_i demandés, tous les paramètres d'entrée demandés des services abstraits candidats sont ajoutés à V et toutes les arrêtes correspondantes (RP_i, RP_k, AS_k, R_k) sont ajoutées à E, sinon, RP_i sera supprimé du graphe car aucune solution n'existe. La génération du graphe G s'arrête lorsque tous les paramètres demandés sont identifiés ou lorsque la composition est impossible.

Deuxième couche : génération du plan optimal

Cette couche utilise le plan global généré dans la première couche d'une part pour la production d'un plan abstrait optimal de la composition et, d'autre part, pour la régénération d'un nouveau plan optimal en cas d'échec.

Une fois que le plan global est obtenu, un plan abstrait optimal de composition est choisi en fonction de la réputation et de la complémentarité des paramètres fournis par les services abstraits candidats.

Cette sélection permet d'éviter l'invocation de services qui fournissent les mêmes paramètres. Par conséquent, le plan optimal généré a le plus petit nombre de services et de paramètres qui peuvent apparaître dans la composition.

- **Estimation de la réputation**

$$RE_t(a) = \frac{NBsucces}{NBselection} \tag{2.4}$$

Où :

RE_t(a) : la réputation du service abstrait a.

La sélection des services se fait selon le profil de l'utilisateur(préférences, localisation. etc), le contexte de l'environnement, et les paramètres de qualité indiqués dans le tableau suivant :

Paramètre	Qualité du paramètre
Temps de réponse (TR)	$\frac{1}{TR}$
Fiabilité	$\frac{NBDonnesReues}{NBDonnesPromise}$
Disponibilité	$\frac{NBDisponible}{NBDomand}$
sécurité	Degré de sécurité

TABLE 2.2 – Paramètres de qualité services.

L'estimation de la qualité d'expérience de chaque service concret se fait comme suit :

$$QoE_t(cs) = \sum_{i=1}^{NbQ} (W_k^i * QoS_t^i(s)). \tag{2.5}$$

Avec :

$$W_k^i = \frac{(WUP_k^i + WUC_k^i + WEC_k^i)}{N} \tag{2.6}$$

Où :

QoE_t(s) :qualité i du service concret S à l'instant t.

NbQ : le nombre de paramètres de qualité de service.

W_k^i : le poids de la qualité i pour l'utilisateur k.

WUP_k^i : le poids de la qualité i pour l'utilisateur k en fonction de ses préférences.
WUC_k^i : le poids de la qualité i pour l'utilisateur k en fonction de son contexte.
WEC_k^i : le poids de la qualité i pour l'utilisateur k en fonction du contexte de son environnement.
N : le nombre d'informations collectées.

Pour chaque service abstrait du plan abstrait, le service concret qui maximise la fonction QoE suivante sera sélectionné :

$$cs_k = ArgMax(QoE_t(x))_{x \epsilon AS_j^{cs}}. \tag{2.7}$$

Troisième couche : surveillance du plan d'exécution

Cette couche est responsable de surveiller le plan d'exécution et de l'adapter automatiquement en cas de défaillances.

2.2.3.2 Discussion

Dans cette solution les auteurs se sont beaucoup intéressés sur l'aspect dynamique des environnements ubiquitaires. Ils ont proposé une approche de composition de services en couches, flexible et tolérante aux pannes qui prend en considération le contexte d'utilisation et la qualité des services disponibles.

Les auteurs proposent d'estimer la qualité d'un service concret à partir de son expérience. Dans un environnement ubiquitaire, deux exécutions du même service peuvent se dérouler à deux instants différents et dans des contextes distincts. Pour cette raison, il semble parfois inadéquat de se reposer sur les dernières exécutions pour estimer le comportement que le service aura. Il serait plus intéressant que les paramètres de QdS ne soient pas estimés à partir des exécutions passées mais à partir du contexte courant.

2.3.4 Modèle de composition de services à base des MDP dans les environnements ubiquitaires

2.2.4.1 Déscription

Dans cette article [32] les auteurs ont proposé une solution pour la sélection et la composition horizontale de services en se basant sur les MDP (Malkov Decision Processes) et les modèles d'apprentissage

bayesiens et en tenant en compte de la probabilité de réponse et de la qualité de service.

Les auteurs proposent deux principaux processus de composition :

La composition verticale : consiste à définir un plan approprié de services pour effectuer la tâche de composition.

La composition horizontale : consiste à déterminer le service le plus approprié parmi l'ensemble fournissant la même tâche.

1. Probabilité de réponse

Un service concret est une tache qui peut être réalisée dans un environnement incertain. Une fois qu'une action est effectuée, deux types de réponses peuvent se présenter : positive avec une probabilité p et négative avec une probabilité q=1-p.

- ◇ **Réponse positive :** le service répond avec tous les paramètres de sorties requis, cela signifie qu'après l'exécution du service toutes les valeurs seront assignées à ses sorties. Ces valeurs représentent l'effet de l'exécution des services.
- ◇ **Réponse négative :** le service ne répond pas avec tous les paramètres de sortie nécessaires, plusieurs raisons peuvent être à l' origine de cette situation (dégradation de la qualité de service).

Dans ce modèle les probabilités représentent l'estimation de réponse pour chaque action, cette estimation initiale peut être imprécise ou complètement inconnue c'est pourquoi l'introduction d'un mécanisme d'apprentissage devient nécessaire. Dans cette solution les auteurs ont utilisé l'apprentissage bayésien.

2. Apprentissage bayésien

L'algorithme d'apprentissage bayésien maintient un compteur d'expérience noté expr initialisé à 1 pour chaque action. A l'exécution d'une action a, on obtient une réponse positive ou négative.

La probabilité de l'action a s'obtient de la manière suivante :

$$p' = \frac{[p * expr] + 1}{expr + 1}(rponsepositive). \tag{2.8}$$

$$q' = 1 - p'(rponsengative). \tag{2.9}$$

3. Estimation de la probabilité de réponse

L'incertitude globale sur l'invocation d'un service concret s est estimée par la probabilité P_t (s) selon la formule suivante :

$$P_t(s) = RT_t(s) * Re_t(s) * Av_t(s). \tag{2.10}$$

Où :

RT_t(s) : la valeur estimée du temps de réponse du service concret s.

Re_t(s) : la valeur estimée de la fiabilité du service concret s.

Av_t(s) : la valeur estimée de la disponibilité du service concret s.

4. Estimation de la qualité de service

La qualité globale du service concret s est estimée comme suit :

$$QoS_t(s) = \sum_{i=1}^{NbQ} (W_k^i * QoS_t^i(s)). \tag{2.11}$$

Où :

QoS_t^i(s) :qualité estimée du paramètre i du service concret s à l'instant t.

NbQ : le nombre de paramètres de qualité de services.

W_k^i(s) : le poids de la qualité i du service concret s pour l'utilisateur k.

La qualité de service estimée combinée avec la probabilité de réponse positive est utilisée comme un mécanisme de sélection du meilleur service concret et cela en utilisant la formule suivante :

$$cs = ArgMax(P_t(s) * QoS_t(s)). \tag{2.12}$$

2.2.4.2 Discussion

La dynamicité des environnements ubiquitaires pose un problème lorsqu'un ou plusieurs services qui font parties d'une composition de ces derniers ne sont plus disponibles ou bien que leur QdS diminue lors de la composition.

Les auteurs de cette approche ont proposé une solution pour la composition horizontale de services basée sur le modèle MDP et l'apprentissage bayésien , qui fait face à la dynamicité des environnements ubiquitaires.

2.3.5 Sélection des périphériques avec QoS en utilisant les préférences de l'utilisateur dans un environnement ubiquitaire

2.2.5.1 Déscription

Les approches existantes de composition de services sont principalement limitées au service correspondant et considèrent les aspects fonctionnels de la combinaison seulement.

Dans [33], les auteurs ont proposé une sélection des dispositifs sensibles à la QdS dans le milieu omniprésent afin de maintenir un équilibre entre la QdS et les préférences de l'utilisateur. Lors de la sélection d'un service, ces capacités doivent être considérées et exprimées en terme de QdS des dispositifs, la QdS des réseaux et les préférences de l'utilisateur.

QdS des dispositifs : les dispositifs ayant une meilleure CPU et un espace mémoire suffisant et ceux qui répondent aux besoins de l'utilisateur d'une manière efficace seront sélectionnés.

QdS des réseaux : le but est de sélectionner un lien qui a un taux de réussite maximale avec un temps de latence minimal et qui offre un haut débit.

Dans cette approche, les auteurs ont défini un services composite comme étant une tâche qui est composée de plusieurs services T=S1,S2..,Sn. La sélection se déroule selon les trois étapes suivantes :

Etape(1) : Découverte des périphériques

Pour chaque service de la tâche T, on aura un ensemble de dispositifs sur lesquels le service peut s'exécuter comme le montre la figure suivante :

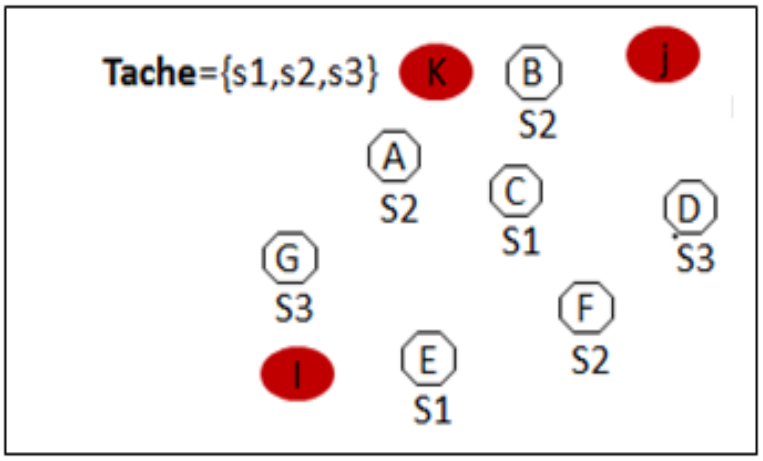

FIGURE 2.8 – Découverte de périphériques.

Etape(2) : Découverte des périphériques sensibles à l'hétérogénéité

Vu que les services résident dans des dispositifs différents, il faut qu'il ait une interface de communication commune entre les paires de service, si l'un des dispositifs est isolé, il sera ignoré comme il est illustré dans la figure suivante :

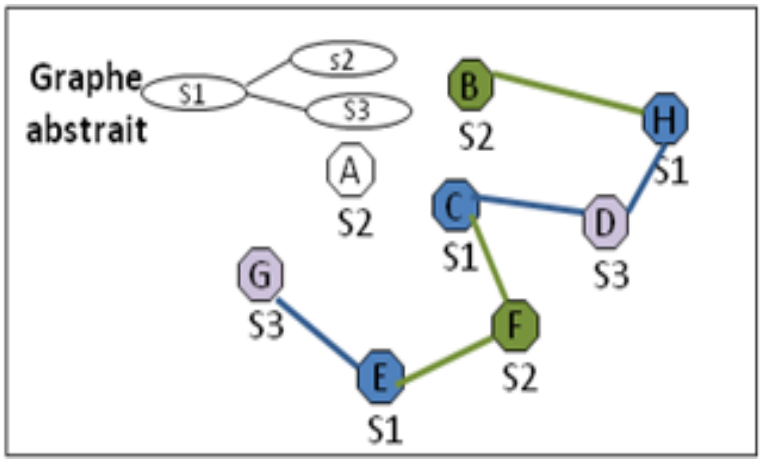

FIGURE 2.9 – Découverte de périphériques sensibles à l'hétérogénéité.

Etape(3) : Sélection des périphériques

Après avoir calculé les QdS des réseaux ainsi que les QdS des dispositifs, celles qui maximisent les préférences de l'utilisateur seront sélectionnées, voir la figure suivante :

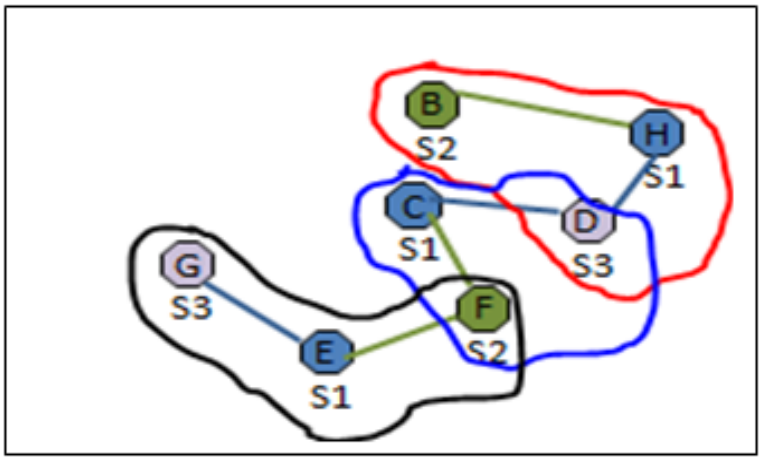

FIGURE 2.10 – Sélection des périphériques.

2.2.5.2 Discussion

En cas de panne ou absence du service participant à la composition du service composite, on refait complètement la découverte des services. Il devient plus intéressant de refaire la découverte au niveau où la panne survient afin de remplacer le service en panne.

La dynamicité de ces milieux de services pose certains problèmes, lorsqu'un ou plusieurs services qui font parties d'une composition de services ne sont plus disponibles ou bien que leur QdS connait une baisse (dû par exemple à une déconnexion ou la connectivité faible du réseau) lors de l'exécution de la composition. Ainsi, un service sélectionné, pour participer à une composition basée sur la QdS peut ne plus fournir la même QdS énoncée lorsque vient le temps d'être réellement invoqué.

2.3.6 Sélection et composition de services automatique sensible aux événements dans les environnements ubiquitaires

2.2.6.1 Description

Dans cette solution [34], les auteurs ont proposé une approche à base de règles pour la sélection et la composition automatique de services qui se déroule en deux phases : la phase hors ligne et la phase en ligne.

La phase hors ligne : un graphe global qui relie tous les services abstraits disponibles est généré automatiquement et cela en utilisant une technique à base de règles.

La phase en ligne : dans cette phase un sous graphe est extrait a partir du graphe global selon l'événement survenu et détecté dans l'environnement.

Règles de composition

Les règles de composition visent à optimiser à la fois le nombre de services et le nombre de paramètres qui apparaissent dans le graphe global. Il existe cinq règles :

1. **Règle de marquage**

   Elle élimine la confusion sur les sources de tous les paramètres qui participent à la composition de services et réduit leurs nombre total en ne conservant que les services pour lesquels au moins un paramètre de sorti est utilisé dans le graphe de composition. Tous les paramètres et leurs sources correspondantes dans le graphe de composition sont conservés dans un tableau appelé **tableau de marquage**.

2. **Règle d'égalité**

   Pour deux services abstraits qui apportent les mêmes valeurs a la composition, les auteurs ont défini une règle qui supprime le plus grand service en terme de paramètres d'entrés. Ce choix est motivé par le fait qu'un service qui possède plusieurs paramètres d'entrés sera probablement lié à plusieurs autre services dans la composition. Par conséquent la suppression de ce service provoque probablement la suppression de certains autres services ainsi le nombre de service sera minimisé. Dans le cas ou les services possèdent le même nombre de paramètres d'entrés, le service abstrait qui a le plus petit nombre de service concret sera supprimé.

3. **Règle d'inclusion simple**

   Si les sorties d'un service abstrait sont inclus dans les sorties d'un autre. Ce service doit être supprimé du graphe car il n'apporte rien à la composition.

4. **Règle d'inclusion complexe**

   A chaque niveau du graphe, un service abstrait candidat sera supprimé si tous ses paramètres de sorties sont déjà livrés (marqués).

5. **Règle de démarquage**

   Elle consiste à supprimer de la table de marquage tous les services abstraits qui ne participent pas à la composition.

Phase hors ligne (construction du graphe global)

Pour la phase hors ligne, les auteurs ont proposé un algorithme qui construit un graphe global qui possède plusieurs niveaux appelé graphe abstrait de la composition de service (AGoSC) qui relie les services abstraits disponibles en utilisant de manière appropriée les règles suivantes : la règle d'inclusion simple, la règle d'égalité, la règle d'inclusion complexe et la règle de marquage.

Phase en ligne (construction du sous graphe)

Pour la phase en ligne, les auteurs ont proposé un algorithme qui extrait un sous graphe appelé graphe de composition de services (SubGoSC) à partir du graphe global obtenu dans la phase hors ligne et de l'événement détecté dans l'environnement.

L'exécution de l'algorithme commence a partir du dernier niveau du graphe AGoSC jusqu'au au premier niveau en explorant l'ensemble des niveaux intermédiaires. A chaque niveau du graphe, la règle de démarquage est appliquée afin de ne conserver que les services qui fournissent au moins un paramètre parmi les paramètres demandés.

Modèle de sélection de services

- **Estimation de la qualité de service statique**

  La qualité statique globale obtenue lors de la réalisation du service concret cs est noté (SQoS)(cs) tels que SQoS(cs)ϵ[0,1] . Elle est estimé par les formules (2.10) ou (2.11).

$$SQoS(cs) = \frac{\sum_k W_k * (N - CS_{s,k})}{N * \sum_k W_k} \tag{2.13}$$

$$SQoS(cs) = \frac{1 - \sum_k W_k * CS_{s,k}}{N * \sum_k W_k} \tag{2.14}$$

  Où :

  N : la position du service concret cs.

  W_k : le poids associé au paramètre k selon les préférences de l'utilisateur.

- **Estimation de la probabilité de réponse**

  La probabilité de réponse dépend principalement de la disponibilité (AV), la fiabilité (RE), le temps de réponse (RT) et du délai (timeOut) que le service concret ne doit pas dépasser à répondre avec tous ses paramètres de sorties requis. Elle est estimée en utilisant la formule suivante :

$$PR_t(cs) = AV_t * (1 - \frac{RT_t(cs)}{timeOut + \varepsilon}) * RE_t(cs) \tag{2.15}$$

  Si la probabilité de réponse PR d'un service concret connait une baisse (du par exemple à une déconnection ou la connectivité faible du réseau) à l'instant t , nous ne pouvons pas en déduire sa probabilité de réponse a l'instant t+1. Pour cette raison il faut tenir en compte de l'historique des probabilités de réponse afin de sélectionner le meilleur service concret.

  L'historique des probabilités de réponse est calculé selon la formule suivante :

$$HPR_t(cs) = HPR_{t-1}(cs) + PR_t(cs) \tag{2.16}$$

- **Estimation de la qualité de service dynamique**

  qualité dynamique dépend principalement de l'historique des probabilités de réponse HPR et le nombre de requêtes NR à l'instant t. Elle est calculée comme suit :

$$QDoS_t(cs) = \frac{HPR_t(cs)}{NR_t(cs)} \tag{2.17}$$

La sélection du meilleur service concret prend en compte à la fois sa qualité de service statique à l'instant t et sa qualité de service dynamique à l'instant t-1 et selon la formule suivante :

$$Selection(cs) = ArgMax(SQoS(cs) * DQoS_{t-1}(cs)) \tag{2.18}$$

2.2.6.2 Discussion

Les auteurs ont proposé une approche de composition de services dynamique et tolérante aux échecs pour les environnements ubiquitaires et qui prend en considération les préférences et les exigences de l'utilisateur sous forme de paramètres de qualité de service.

Dans la sélection du service concret les auteurs ont proposé de tenir compte de sa qualité dynamique obtenu de son invocation précédente. Afin d'estimer le comportement qu'un service concret aura, il serait plus intéressant que sa qualité de service dynamique ne soit pas estimé à partir des dernières exécutions mais prédit à partir de son exécution courante.

2.3.7 Sélection flexible de services avec QoS en se basant sur l'architecture orientée service

2.2.7.1 Désrciption

Dans cette solution [35], les auteurs ont proposé un algorithme de sélection de services pour les systèmes SOA. L'algorithme propose un certain nombre de services appropriés en fonction des exigences de l'utilisateur en termes de qualité de service.

Le modèle de composition de services se fait en trois parties :

- **Soumission du travail**

  La requête du travail de l'utilisateur est décrite par un modèle et elle est soumise à la composante de soumission de travail, qui est en charge de formaliser et de stoker les données d'entrée. Après avoir déterminé le prestataire de services par le planificateur, la composante de la soumission de travail envoie les données d'entrée au prestataire de services.

- **Planificateur**

  Il est en charge de communiquer avec le registre UDDI, pour chercher et planifier le prestataire de services approprié à la requête de l'utilisateur. Il a les trois fonctionnalités suivantes :

  1. Les composantes chargées de traiter la requête de l'utilisateur communiquent avec le registre UDDI pour trouver le prestataire de service approprié, et calculent les valeurs de qualité de service.
  2. Sélectionner les prestataires de services candidats et appliquer l'algorithme SPSE.
  3. Obtenir la décision finale de l'exécution de l'utilisateur et met à jour ses préférences.

- **Moniteur d'exécution**

  Il est chargé de surveiller l'exécution du plan correspondant à la requête de l'utilisateur. Si le prestataire de services sélectionné n'existe plus il activera le planificateur afin de trouver un autre prestataire de services pour exécuter la requête demandée. Apres que l'exécution soit terminée, il est également en charge de collecter les résultats d'exécution.

Les services qui répondent aux paramètres de qualité de service tels que : le degré de confiance, le temps de réponse, de coût monétaire, et la plate-forme / API pourraient éventuellement être sélectionnées pour l'exécution. Par conséquent, la performance du workflow est agrégée de la performance respective de chaque tâche. Le degré de confiance global et le coût monétaire sont obtenus de la manière suivante :

$$DegreDeConfiance_{workflow} = \prod_{i=1}^{n} DegreDeConfiance_i \tag{2.19}$$

$$Cout_{workflow} = \sum_{i=1}^{n} Cout_i \tag{2.20}$$

où :

n : représente le nombre de tâches dans le workflow.

Le temps d'exécution du workflow peut être estimé comme suit :

$$T_{workflow} = \sum_{t \epsilon workflow} T_i \tag{2.21}$$

la tâche t appartient au workflow et détermine la longueur du temps d'exécution globale.

L'algorithme de planification de service

L'algorithme de planification se comporte de quartes instructions :

1. **SPL$_i$← Chercher (travail$_i$) :** sert à chercher et renvoyer une liste de prestataires de services qui satisfont la requête de travail$_i$.

2. **SoL$_i$← Filtrer (SPL$_i$) :** sert à supprimer de la liste SLP$_i$ tous les prestataires qui ne satisfont pas les exigences de l'utilisateur..

3. **RSoL$_i$← Classer (SoL$_i$) :** sert à classer les meilleurs prestataires de services en termes de QoS.

4. **UP← Mettre-à-jour(UP) :** : l'utilisateur sélectionne un prestataire de services de la liste RoL$_i$ et son travail$_i$ s'exécute selon son choix. En se basant sur la sélection, mettre-à-jour(UP) est l'instruction chargée de calculer les préférences de l'utilisateur.

2.2.7.2 Discussion

L'approche proposée porte sur l'algorithme de planification SPSE pour la composition flexible de service avec QoS en se basant sur l'architecture orientée service.

La sélection des services se fait uniquement selon les paramètres de qualité de services et selon les exigences de l'utilisateur, sans aucune prise en considération des informations contextuelles.

2.3.8 Construction des applications omniprésentes sensibles aux QoC en utilisant un modèle basé sur le génie logiciel

2.2.8.1 Désrciption

Les auteurs de cette solution [36] ont proposé un processus de conception générique et sensibles au contexte pour les applications, en tenant compte de la qualité des informations contextuelles QoC (Quality of Context). Ils ont développé la conception de la partie du processus de gestion du contexte PCM (Processus-oriented Context Manager) et la conception de la sensibilité au contexte d'applications omniprésentes sur les appareils mobiles en proposant un processus de manipulation de QoC, l'objectif de cet article est de mettre en évidence les exigences OoC et les prendre en considération lors de l'utilisation d'un service.

Intégration de QoC dans le processus de conception d'applications sensibles au contexte

Le rôle du gestionnaire de contexte est de fournir des données contextuelles de haut niveau pour identifier les situations qui peuvent provoquer des réactions dans les applications sensibles au contexte afin d'acquérir, de traiter et de présenter ces données aux applications utilisées dans l'environnement.

Ils ont augmenté le domaine d'application d'entreprise classique avec deux domaines spécialisés :

- **Gestionnaire du contexte :** Il assure quatre principales activités :
    1. **Spécification des collectionneurs de contexte :** Par exemple, le concepteur peut mettre en œuvre plusieurs collectionneurs de contexte pour obtenir des valeurs de localisation de différents capteurs GPS, GPRS ou Wifi et par différents dispositifs des téléphones Androïde, i Phone ou des téléphones Windows Mobiles.

2. **Conception des opérateurs de contexte :** correspond à la transformation du contexte, la particularité de cette approche réside dans les opérateurs de contexte qui sont complétés par le calcul de la QoC.

3. **Conception des paramètres de QoC et des méta-données :** Il sert à surveiller par exemple, la fraîcheur des l'informations contextuelles.

4. **Concevoir les entités qui modélisent les données contextuelles de haut niveau :** En écrivant dans la COSMOS DSL les expressions des traitements du contexte. Qui réutilisent les collectionneurs de contexte, les opérateurs de contexte, et les opérateurs de paramètres QoS conçus dans les activités précédentes.

- **Concepteur de sensibilisation au contexte :** Il assure les principales activités suivantes :

1. Spécification des entités qui doivent être observées et observables par exemple, dans le cas de l'application de la vente flash, le client et les commerçant sont des exemples d'entités à observer, et l'emplacement et les produits préférés sont des exemples de facteurs observables.

2. Spécification d'un contrat pour lier la gestion de contexte de travail des produits ou politiques de contexte, Par exemple, un contrat précise que le service de vente flash doivent être déclenchés quand une vente flash intéressante est détectée.

Le modèle d'approche pour la sensibilité au contexte avec QoC

Ils suivent une approche pilotée par des modèles afin de définir la sensibilité au contexte de l'application par un modèle conforme aux méta-modèles, ils se sont concentrés sur le traitement de la QoC dans le processus de gestion de sensibilité au contexte.

Un Framework définit un contrat pour une donnée observable et une application donnée à remplir par le service de gestion de contexte, ce contrat définit toutes les exigences de QoC ainsi que le niveau de QoC de chaque attribut de paramètres de contexte, ils ont pris en considération trois contrats qui sont : (i)contrat d'observation : quand un concepteur de l'application veut observer et synchroniser les données contextuelles ; (ii)contrat de notification : le concepteur de l'application peut spécifier des abonnements à des événements qui se produisent , et (iii) un contrat d'adaptation : met en action les décisions d'adap-

tation prises par la demande suite à la détection d'une situation d'adaptation spécifique, afin de fournir le meilleur service par rapport aux QoC.

2.2.8.2 Discussion

Les paramètres de QoS n'ont pas été pris en compte dans cette approche, cela ne reflète pas les réels besoins et exigences de l'utilisateur. Cette approche s'est focalisée sur l'aspect dynamique de l'environnement ubiquitaire et l'utilisation des applications mobiles, elle considère notamment les qualités des informations contextuelle par l'emploi des méa-données afin de fournir les différents services disponibles convenables à la situation de l'utilisateur.

2.3.9 Approche adaptative à base de buts pour la composition de services en utilisant la planification

2.2.9.1 Désription

La planification de réseau des taches hiérarchiques HTN (Hierarchical Task Network) est très prometteuse parce que le concept de décomposition des tâches dans la planification HTN est très similaire à la notion de décomposition de processus composite dans une approche basée sur les buts pour atteindre l'état final que l'utilisateur a décrit , en utilisant le concept de but comme exigence de l'utilisateur.

Dans [37], les auteurs ont proposé un système qui offre plusieurs avantages dans la composition de services :

- **Amélioration de la convivialité :** permet aux utilisateurs de spécifier leurs objectifs d'une manière simple ;

- **Adaptable aux utilisateurs :** Le système offre un service personnalisé pour les utilisateurs ;

- **Réutilisation dans un autre domaine :** le service peut être facilement réutilisé et étendu à d'autres domaines.

Ils ont développé un prototype préliminaire qui prend en charge la composition de services dans un environnement de travail intelligent par ce que les applications et les périphériques peuvent échouer pour plusieurs raisons. Par conséquent, ils ont exigé des techniques pour simplifier les environnements afin de

permettre aux utilisateurs d'atteindre leurs buts.

Ensuite le Framework analyse les différents choix disponibles pour la réalisation des buts des utilisateurs et sur la base de leurs préférences, leurs besoins, le contexte actuel et les politiques de contrôle d'accès, le Framework choisit la meilleure façon d'atteindre le but ; il prévoit ensuite une séquence d'actions pour l'atteindre et l'exécuter.

Présentation du système

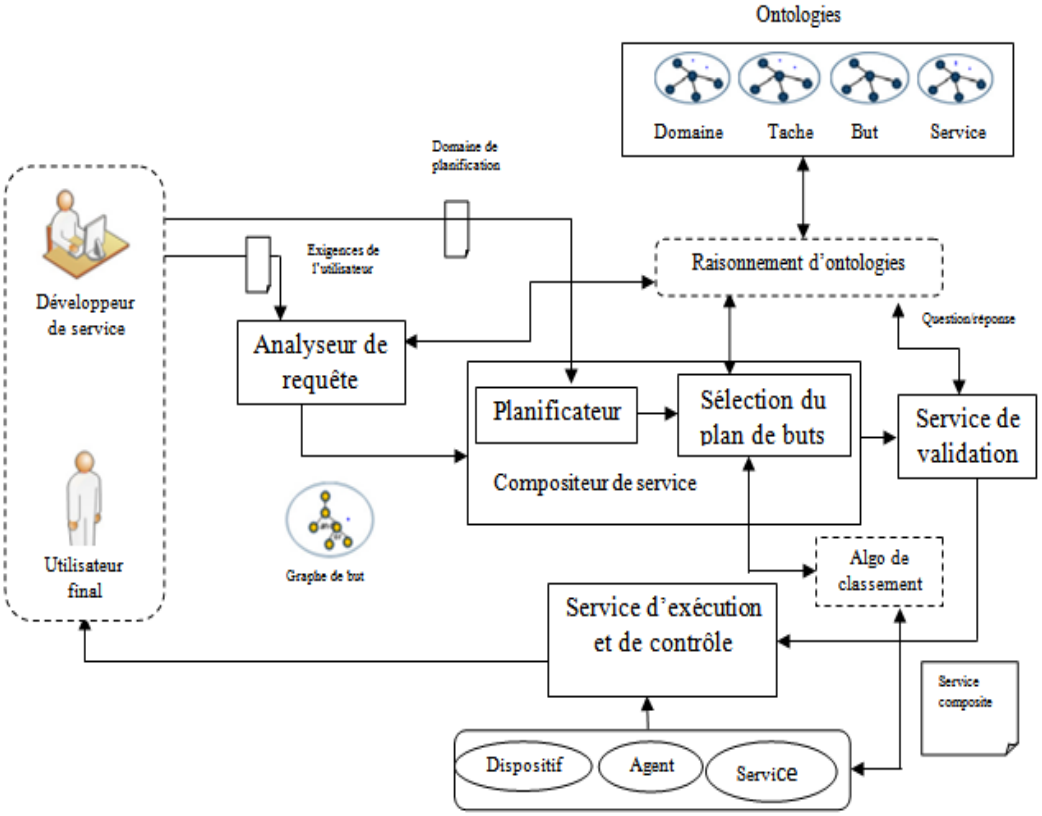

FIGURE 2.11 – Présentation du système de composition de services.

Les auteurs ont conçu un système de composition de services pour maintenir la composition dynamique de services comme le montre la Figure 2.11, le système se compose de quatre éléments principaux :

•**Analyseur de requête :** Se charge de la réception des requêtes de l'utilisateur et les analyser en utilisant le raisonnement basé sur les ontologies qui fournira comme résultat un graphe de buts qui se compose d'un but principal et des sous buts.

•**Planificateur HTN :** Après avoir construit un graphe de but, le planificateur HTN fait un plan pour la réalisation de chaque sous-but. Au cours de ce processus, le planificateur décompose chaque sous-but dans un réseau de tâches hiérarchisées.

•**Compositeur de services :** Après la construction du plan pour chaque sous-but, le compositeur de services sélectionne le plan le plus approprié en utilisant un algorithme basé sur le classement, et les combines en un nouveau service composite.

•**Service de validation :** Il vérifie les résultats de la composition de services afin de l'exécuter.

2.2.9.2 Discussion

Les auteurs ont proposé une approche de composition de services adaptative basée sur la planification tout en prenant en considération le contexte d'utilisation et la qualité des services disponibles. Cette approche s'est basée sur la prise en compte des besoins des utilisateurs comme buts.

Durant la composition, si un ou plusieurs services faisant parties de la composition ne sont plus disponibles, le système résout ce problème en se basant sur un raisonnement d'ontologies pour la sémantique.

2.4 Synthèse et comparaison entre les approches étudiées

La composition de services dans les environnements ubiquitaires est un domaine de recherche inspiré des recherches sur les services web. Plusieurs approches ont été proposées pour résoudre ce problème, et la plupart d'entres elles visent à intégrer l'automatisation du processus de composition des services, la prise en compte de la nature dynamique des environnements ubiquitaires, et intégration de la sensibilité au contexte. C'est dans cette idée que s'inscrivent les travaux de recherche menés par les auteurs des approches étudiées précédemment.

Chacune de ces approches présente des points forts dans certains aspects de sélection et de composition mais aussi quelques lacunes. Les approches déjà présentées sont centralisées autour d'un moteur de sélection et de composition, elles présentent des aspects importants qui s'adaptent à la nature imprévisible et instable de l'environnement ubiquitaire (considéré comme ouvert, partiellement connu et

fortement dynamique). Cette nature est d'une part le dynamisme dans la mesure où les entités qui le composent sont mobiles et susceptibles de tomber en panne, et dans la composition cela se traduit par une sélection combinant les services de manière appropriée selon la requéte de l'utilisateur. Et d'autre part l'aspect automatique, dans le sens où la sélection et la logique d'interconnexion entre les services se fait d'une manière implicite sans l'intervention de l'utilisateur.

Les approches passées en revue pour la sélection et la composition se différent par la techniques utilisées, leur degré d'automatisation, leur prise en compte de la sémantique (basées sur les ontologies ou non), du contexte, et de la QoS, également par leur caractéristique dynamique ou statique.

Le tableau suivant résume les différentes caractéristiques des approches de composition de services, qu'on a passés en revue dans les environnements ubiquitaires :

Approche	Référence	Principe utilisé	caractéristiques
La combinaison de l'optimisation globale et la sélection locale pour une composition de service avec QoS.	[25]	Programmation entière mixte (MIP) pour la décomposition des contraintes de QoS globales en contraintes locales.	Seuls les attributs négatifs sont considérés. Seul le modèle séquentiel est utilisé.
Composition de services avec QoS dans les environnements dynamiques orientés services.	[26]	L'algorithme K-means de clustering.	la prise en compte des contraintes globales de QdS.
Composition de services sensible au contexte dans les environnements ubiquitaires.	[31]	Mécanisme de conception en couches.	Prise en compte des informations contextuelles et de la qualité de service lors de la sélection et de la composition de services.
Modèle de composition de services à base des MDP dans les environnements ubiquitaires.	[32]	Modèle d'apprentissage bayesiens et les MDP.	Prise en considération de la probabilité de réponse et de la qualité de service lors de la sélection de services.
Sélection des périphériques avec QoS en utilisant les préférences de l'utilisateur dans les environnements ubiquitaires.	[33]	Intégration d'un support de QoS dans les intergiciels.	L'équilibrage entre la QoS des périphériques et les préférences de l'utilisateur.
Sélection et composition de services automatique sensible aux événements dans les environnements ubiquitaires	[34]	Algorithme de composition de services à base de règles.	Prise en compte de la qualité de service et du contexte d'utilisation.

Approche	Référence	Principe utilisé	Caractéristiques
Sélection flexible de services avec QoS en se basant sur l'architecture orientée service	[35]	Algorithme de planification SPSE.	Prise en compte de la qualité de service et des exigences de l'utilisateur.
Construction des applications omniprésentes sensibles aux QoC en utilisant un modèle basé sur le génie logiciel.	[36]	Les méta-données.	La prise en considération des informations contextuelles.
Approche d'adaptation pour la composition de services à base de buts.	[37]	La planification de réseau des taches hiérarchiques HTN.	La prise en compte du contexte de l'utilisateur et de la QdS.

TABLE 2.3 – Tableau comparatif.

2.5 Conclusion

Dans ce chapitre, nous avons commencé par citer un ensemble de définitions relatives des méthodes de composition de services puis nous avons fait une synthèse concernant la problématique de sélection et de composition de services à travers certaines approches présentées dans la littérature. Le chapitre suivant sera consacré à présenter notre approche de sélection et de composition de services dans les environnements ubiquitaires.

Chapitre 3

Approche de sélection et de composition de services

3.1 Introduction

Le chapitre précédent nous a permis d'avoir une vision générale sur le problème de composition et de sélection de services dans les environnements ubiquitaires en se basant sur différentes techniques et divers principes.

La sensibilité au contexte et la qualité de service sont deux facteurs important dans la sélection des meilleurs services composants qui participent à la réalisation du service composite.

Dans ce chapitre, nous allons mettre l'accent sur notre proposition concernant la problématique citée au dessus, pour cela nous allons présenter en premier lieu, l'importance de la qualité de service et de la sensibilité au contexte, en deuxième lieu nous allons détailler notre contribution.

3.2 Définition des services

3.2.1 Service concret

Un service concret est une action qui agit sur ses entrées pour fournir des résultats en sortie, après son exécution [21]. Un service concret noté CS_i est décrit par un tuple : (CS_i^{in}, CS_i^{out}, Prec, Eff, C, QoS) tel que :

CS_i^{in} : sont les paramètres d'entrées du service ;

CS_i^{out} : sont les paramètres de sortie du service ;

Prec : les pré-conditions du service ;

Eff : les effets du service ;

QoS : les paramètres de qualité du service ;

C : les informations contextuelles du service.

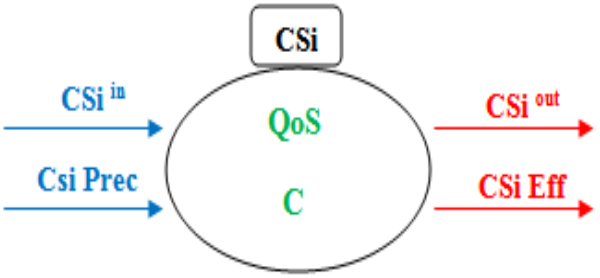

FIGURE 3.1 – Représentation d'un service concret.

3.2.2 Service abstrait (AS)

Un service abstrait est une représentation d'un ensemble de services concrets qui assurent les mêmes fonctionnalités. Ces services concrets ont les mêmes paramètres d'entrée et fournissent les mêmes paramètres de sortie [21]. Un service abstrait noté AS_i est décrit par un tuple (AS_i^{in} ,AS_i^{out},CS) tel que :

AS_i^{in} : sont les paramètres d'entrée du service abstrait AS_i ;

AS_i^{out} : sont les paramètres de sortie du service abstrait AS_i ;

CS : l'ensemble des services concrets.

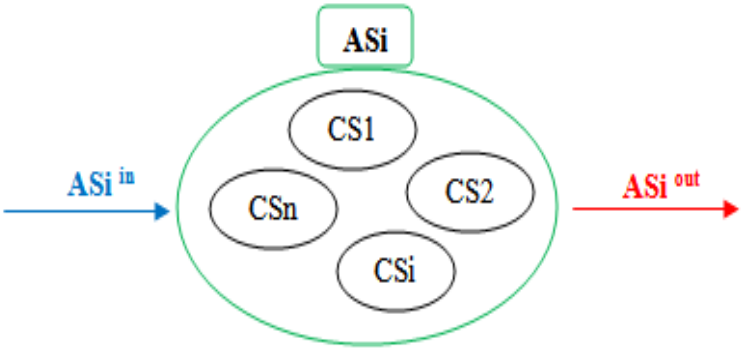

FIGURE 3.2 – Représentation d'un service abstrait.

3.3 Les paramètres de QoS

Les paramètres de la qualité de service (QoS) concernent les paramètres de qualité qui sont spécifiés dans la description des services comme étant des critères de comparaison entre les services qui sont fonctionnellement équivalents. Chaque paramètre nous renseigne sur une valeur d'une qualité bien précise fournie par un service tels que : la sécurité, le coût, le temps de réponse, etc. Ces attributs sont divisés en deux classes : les attributs positifs qui sont les attributs qu'il faut maximiser (disponibilité, fiabilité, sécurité) et les attributs négatifs qui représentent les attributs qu'il faut minimiser (Temps de réponse et le coût). En fonction de la fréquence de leur changement et de mise à jour, nous distinguons deux types de paramètres de qualité de service : les paramètres de qualité statique (SQP : Static Quality Parameters) et les paramètres de qualité dynamique (DQP : Dynamic Quality Parameters) [38].

3.3.1 Les paramètres de qualité statique (SQP)

Il s'agit des paramètres de qualité qui restent inchangés pendant une longue période de temps. En d'autres termes, la fréquence de mise à jour des valeurs de ces paramètres est très petite. Parmi ces paramètres, on peut citer à titre d'exemple, le niveau de sécurité d'un service, le coût d'un service, etc.

1. **Sécurité :** indique le niveau de sécurité assuré par le service.
2. **Coût :** représente les frais générés par l'utilisation d'un service (prix d'une prestation de service).

3.3.2 Les paramètres de qualité dynamique (DQP)

Contrairement aux paramètres de qualité statique, les paramètres de qualité dynamique changent en fonction du contexte d'utilisations courant. Dans ce qui suit, nous considérons trois paramètres de qualité dynamique :

1. **Disponibilité(AV) :** Le service devrait être prêt pour une consommation (exécution) immédiate lors d'une invocation.
2. **Fiabilité(RE) :** La fiabilité est la probabilité que le service répond bien à la requête avec le maximum des données attendues indiqués dans la description du service.
3. **Temps de réponse(RT) :** Le temps de réponse d'un service mesure le temps entre le moment de l'envoi de la requête et le moment où la réponse est reçue.

3.4 Modélisation du contexte

Le contexte et la sensibilité au contexte sont des facteurs très importants dans l'implémentation des applications ambiantes. Il existe quatre catégories de contexte :

3.4.1 Le contexte environnement

Représente toutes les informations qui entourent le système, le degré de la température, le niveau de la luminosité ainsi que le niveau du bruit.

3.4.2 Le contexte utilisateur

Englobe toutes les informations qui concernent l'utilisateur telles que son profil, ces préférences, sa localisation, son activité courante, son état physique,...etc.

3.4.3 Le contexte dispositif

Se sont les informations relatives à l'environnement d'exécution, en d'autres termes, le contexte dispositif concerne les caractéristiques des périphériques utilisés (processeur, mémoire, puissance de traitement,...etc.) de tous les dispositifs disponibles dans l'environnement.

3.4.4 Le contexte service

Concerne toutes les informations non fonctionnelles du service, à savoir son temps d'exécution, son coût, sa disponibilité, sa fiabilité ainsi que sa probabilité de réponse.

3.5 Approche de sélection et de composition de services

Cette approche se focalise sur le problème de sélection et de composition de services en fonction des préférences des utilisateurs en termes de qualité de service (QoS) et du contexte d'utilisation. Cette approche se base sur les hypothèses suivantes :

1. Les services concrets disponibles sont découverts par le module de découverte de services et mis à disposition pour une éventuelle utilisation dans l'annuaire des services concrets.
2. Les services concrets sont classés en fonction de leurs tâches dans les services abstraits.

3. Une mise à jour régulière est effectuée en fonction de la disparition ou de l'apparition des services concrets.

Afin d'apporter une solution aux problèmes de sélection et de composition de services, nous avons proposé une approche intégrant plusieurs étapes à savoir : i) l'étape de construction du plan global abstrait, ii) l'étape de génération du plan optimal, iii) l'étape d'évaluation du contexte et des paramètres de QdS, iv) l'étape de sélection de services et, v) l'étape de surveillance du plan d'exécution.

3.5.1 Construction du plan global

Elle consiste à construire automatiquement un plan global abstrait. Le plan contient tous les services abstraits possibles qui pourraient participé à composer le service demandé exprimé dans la requête de l'utilisateur.

Pour générer le plan abstrait, on utilise un algorithme de planification par chainage arrière afin de produire une chaîne de services pouvant fournir les sorties requises en utilisant les entrées [5]. Plus précisément, les étapes suivantes sont exécutées :

- Sélectionner tous les services abstraits c'est-à-dire AS_1,AS_2,..,AS_n qui fournissent tous ou une partie des paramètres de sortie requis.
- chaque service sélectionné AS_i, une nouvelle sélection est faite pour trouver l'ensemble des services abstraits qui génèrent les paramètres d'entrée de AS_i . Ce processus est itéré jusqu'à ce que les entrées du service AS_{n-x} correspondent aux entrées de la requête.
- Finalement, le workflow qui spécifie l'ordre d'exécution des composants AS_1 jusqu'à AS_n sera formulé.

Une fois le plan abstrait dérivé, son exécution est faite en traversant le graphe de ses extrémités en remontant jusqu'à l'état initial qui représente le paramètre de sortie de la requête de l'utilisateur.

3.5.2 Génération du Plan optimal

Après avoir construit un plan global abstrait, nous utilisons un algorithme d'optimisation afin de minimiser le nombre de services abstraits contribuaient à la composition et l'exécution du service composite. Pour se faire, nous avons défini l'algorithme suivant :

Algorithm 1 Algorithme d'optimisation

Input : ensemble de services abstrait AS

Begin

for each as_i in AS **do**

if ($\exists as_j \epsilon$ AS (i≠j)$\wedge as_i^{out}$=as_j^{out}) $\wedge$ (QoSmax(as_j) > QoSmax(as_i)) **then**

Supprimer as_i de AS

end if

end for

for each as_i in AS **do**

if ($\exists as_j \epsilon$ AS (i≠j)$\wedge as_i^{out} \subset as_j^{out}$) **then**

Supprimer as_i de AS

end if

end for

end

Pour deux services abstraits qui apportent les mêmes valeurs a la composition, le service qui possède la qualité de service maximale (QoSmax) sera gardé dans le plan optimal afin de garantir la sélection du meilleur service concret en terme de QoS. De même si les sorties d'un service abstrait sont inclus dans les sorties d'un autre. Ce service doit être supprimé du plan car il n'apporte rien à la composition.

3.5.3 Evaluation du contexte et des paramètres de QoS

Nous allons définir les préférences de l'utilisateur afin de garantir la satisfaction des contraintes locales lors de la sélection des services concrets et la satisfaction des contrainte globales lors de l'exécution du service composite, et cela en terme de QdS et du contexte d'utilisation.

3.4.3.1 Evaluation de la qualité de service

Dans cette section, notre objectif est d'évaluer la qualité de service afin de vérifier la satisfaction des exigences locales et globales de l'utilisateur en termes de qualité de service.

1. **Evaluation de la qualité de service locale**

   Elle permet de classer l'ensemble des services concrets de chaque service abstrait en termes de leurs QdS et les préférences de l'utilisateur, en passant par les étapes suivantes :

- **Etape(1)**

  Cette étape consiste à normaliser les valeurs des attributs de QoS en les transformant en des valeurs comprises entre 0 et 1 afin de pouvoir les comparées. Pour chaque service abstrait, on représente la QoS de chacun de ses services concrets CS_i en utilisant un vecteur de QoS CS_i=<$q_{i,1}$,..,$q_{i,n}$>.

  Où :

  n : Représente le nombre d'attributs de QoS requis par l'utilisateur ;

  $q_{i,j}$: Représente la valeur de l'attribut de QoS j du service i ;

  La normalisation des valeurs des attributs de QoS se fait par l'application des formules suivantes :

  Attributs négatifs :

$$q'_{i,j} = \begin{cases} \frac{q_j^{max} - q_{i,j}}{q_j^{max} - q_j^{min}}, & si \quad q_j^{max} - q_j^{min} \neq 0 \\ 1, & sinon \end{cases} \tag{3.1}$$

  Attributs positifs :

$$q'_{i,j} = \begin{cases} \frac{q_{i,j} - q_j^{min}}{q_j^{max} - q_j^{min}}, & si \quad q_j^{max} - q_j^{min} \neq 0 \\ 1, & sinon \end{cases} \tag{3.2}$$

  Où :

  $q'_{i,j}$: représente la valeur normalisée de l'attribut de QoS j associé au service concret CS_i , il est calculé en utilisant la valeur actuelle $q_{i,j}$, q_{max} et q_{min} qui se réfèrent respectivement aux valeurs maximales et minimales de l'attribut de QdS j.

- **Etape(2)**

  Afin d'obtenir la valeur de QdS de chaque service concret, nous utilisons la formule suivante :

$$QoS_{i,t}(cs) = \sum_{j=1}^{n} W_j * q'_{i,j} \tag{3.3}$$

  Où :

  $q'_{i,j}$: représente la valeur normalisée de l'attribut de QdS j associé au service concret i ;

  W_j : représente le poids exigé par l'utilisateur à l'attribut de QdS j ;

  $QoS_{i,t}$: représente la QdS du service concret i à l'instant t ;

2. **Evaluation de la qualité de service globale**

   Après avoir évalué la QoS d'un service concret atomique CS, nous passons à l'évaluation de la QoS d'un service composite. Cette dernière a comme objectif de vérifier la satisfaction des différentes

contraintes globales spécifiées par l'utilisateur afin de sélectionner le meilleur service composite. L'évaluation de la QoS du service composite varie selon les paramètres considérés et selon le modèle de la composition [26]. Dans notre approche, nous distinguons trois modèles possibles : séquentiel, parallèle et le modèle en boucle comme montré dans le tableau suivant :

Modèle de composition	Description
Séquentiel	Exécution séquentielle des services
ET	Exécution parallèle des services
Boucle	Exécution itérative des services

TABLE 3.1 – Modèles de composition de services.

Selon le modèle du service abstrait dans le schéma de composition, nous associons à chaque paramètre de qualité considéré une métrique :

Attributs de QdS	Modèle de composition		
	SEQUENCE	ET	BOUCLE
Temps de réponse	$\sum_{i=1}^{n} \mathrm{RT}_i$	$\mathrm{Max}(\mathrm{RT}_j)$	RT*N
Fiabilité	$\prod_{i=1}^{n} \mathrm{RE}_i$	$\prod_{i=1}^{n} \mathrm{RE}_i$	RE_N
Disponibilité	$\prod_{i=1}^{n} \mathrm{AV}_i$	$\prod_{i=1}^{n} \mathrm{AV}_i$	RE_N
Prix	$\sum_{i=1}^{n} \mathrm{P}_i$	$\sum_{i=1}^{n} \mathrm{P}_i$	P*N
Sécurité	$\mathrm{Min}(\mathrm{Sec}_i)$	$\mathrm{Min}(\mathrm{Sec}_i)$	Sec

TABLE 3.2 – Calcul des paramètres de QdS du service composite.

3.4.3.2 Evaluation du contexte

Le contexte est représenté par un ensemble d'informations contextuelles représenté comme suit : Contexte = $\{\{(c_1, \mathrm{val}), \ldots,(c_n, \mathrm{val})\} \epsilon C \}$ tel que :

C est l'ensemble des attributs du contexte. c_i représente un attribut de contexte, tels que la température, la position, hauteur, émotion, temps, batterie etc.

Dans notre approche nous allons prendre en considération deux attributs de contexte à savoir, la localisation qui représente la distance entre l'utilisateur et le service (dispositif) de même pour le niveau de batterie afin de sélectionner le dispositif qui possède plus d'énergie.

La normalisation des valeurs de ces attributs est nécessaire afin de pouvoir sélectionner le meilleur service en fonction du contexte, pour cela nous utilisons les formules suivantes :

Attributs négatifs : minimisation des paramètres (exemple : localisation).

$$c'_{i,j} = \begin{cases} \frac{c_j^{max} - c_{i,j}}{c_j^{max} - c_j^{min}}, & si \quad c_j^{max} - c_j^{min} \neq 0 \\ 1, & sinon \end{cases} \tag{3.4}$$

Attributs positifs : maximisation des paramètres (exemple : niveau de batterie).

$$c'_{i,j} = \begin{cases} \frac{c_{i,j} - c_j^{min}}{c_j^{max} - c_j^{min}}, & si \quad c_j^{max} - c_j^{min} \neq 0 \\ 1, & sinon \end{cases} \tag{3.5}$$

Où :

$c'_{i,j}$: représente la valeur normalisée de l'attribut de contexte j associé au service concret CS_i , il est calculé en utilisant la valeur actuelle $c_{i,j}$, c_{max} et c_{min} qui se réfèrent respectivement aux valeurs maximales et minimales de l'attribut de contexte j.

Afin d'obtenir la valeur du contexte de chaque service concret, nous utilisons la formule suivante :

$$C_{i,t}(cs) = \sum_{j=1}^{n} W_j * c'_{i,j} \tag{3.6}$$

Où :

$c'_{i,j}$: représente la valeur normalisée de l'attribut du contexte j associé au service concret i ;

W_j : représente le poids exigé par l'utilisateur à l'attribut de contexte j ;

$C_{i,t}$: représente la du contexte du service concret i à l'instant t ;

3.5.4 Algorithme de classement

Apres avoir obtenu la valeur du contexte et celle de la QoS de chaque service concret, nous allons calculer la fonction d'utilité F de chaque service en appliquant la formule suivante :

$$F_i = W_1 * QoS_i + W_2 * C_i \tag{3.7}$$

l'algorithme suivant a pour objectif de classer tous les services concrets des services abstrait en fonction de leurs QdS et de leurs contextes suivant un ordre décroissant.

Algorithm 2 Algorithme de classement

Input : ensemble de services abstrait AS

Begin

for each CS in AS **do**

for each parameter of QoS in CS **do**

Normaliser les valeurs des paramètres de QoS avec la formule (3.1) ou (3.2)

end for

Calculer QoS avec la formule(3.3)

for each parameter of C in CS **do**

Normaliser les valeurs des paramètres de C avec la formule (3.4) ou (3.5)

end for

Calculer C avec la formule(3.6)

Calculer F avec la formule(3.7)

end for

Classer CS dans AS en fonction de F avec un ordre décroissant

end

La figure suivante représente un service abstrait après avoir exécuté l'algorithme de classement :

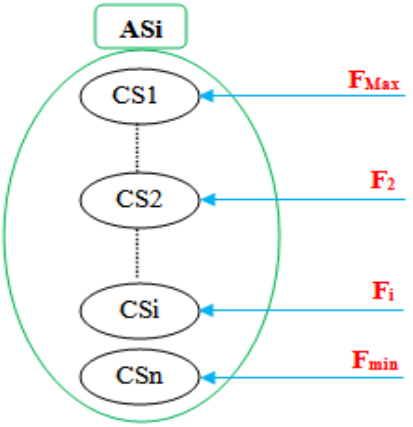

FIGURE 3.3 – Classification des CS dans un AS.

3.5.5 Sélection des services

Dans notre approche, nous considérons deux types de sélections, une sélection locale, appliquée à chaque service abstrait AS, et une sélection globale appliquée sur l'ensemble des services abstraits constituant le service composite (FIGURE 3.4).

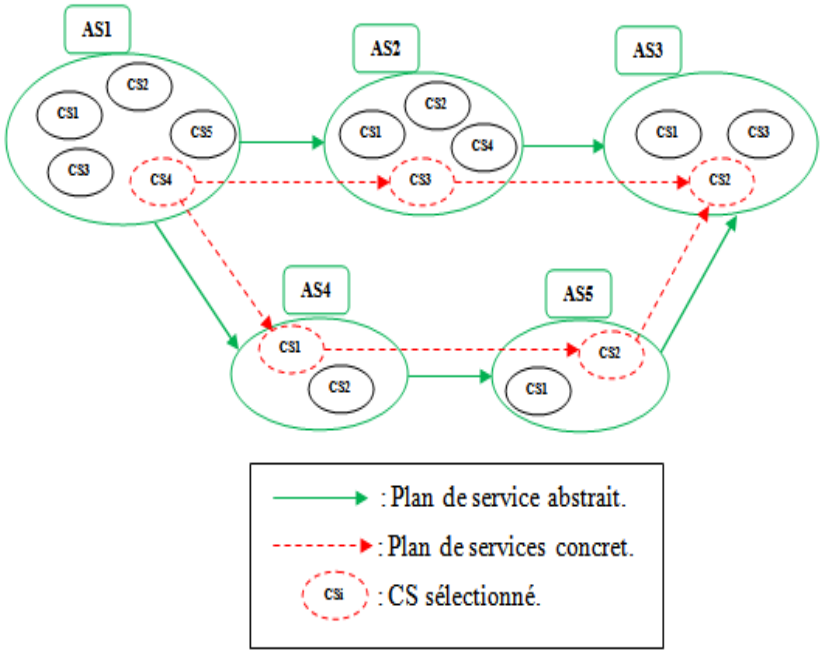

FIGURE 3.4 – Sélection de services.

3.4.5.1 La sélection locale des services

La sélection locale des services consiste à choisir un sous ensemble de services concrets CS de chaque service abstrait AS . Notre approche consiste à définir un seuil T qui représente une borne inferieur de la fonction d'utilité F afin de sélectionner que les services candidats dont leur F>T.

En effet, le nombre de services candidats est réduit ce qui amène à minimiser le nombre de combinaisons possibles c'est-à-dire le nombre de services composites, ainsi que le temps de calcul lors de la composition de services. En d'autres termes c'est de faire face au problème de l'explosion combinatoire.

Vu que l'environnement possède plusieurs services avec des QdS et des contextes différents, le choix du seuil T dépondra de la valeur de la fonction d'utilité maximale (Fmax) et le nombre des services concerts dans chaque service abstrait.

Le seuil T est calculé en utilisant la formule suivante :

$$T = \frac{\sum_{i=1}^{n} F(CS_i)}{n * F_{max}(AS)} \tag{3.8}$$

Où :

$F(CS_i)$: Représente la valeur de la fonction d'utilité du service concret i ;

n : Représente le nombre des service concrets dans le service abstrait ;

$F_{max}(AS)$: Représente la valeur de la fonction d'utilité maximale du service abstrait.

La figure suivante représente l'utilisation du seuil T dans un service abstrait :

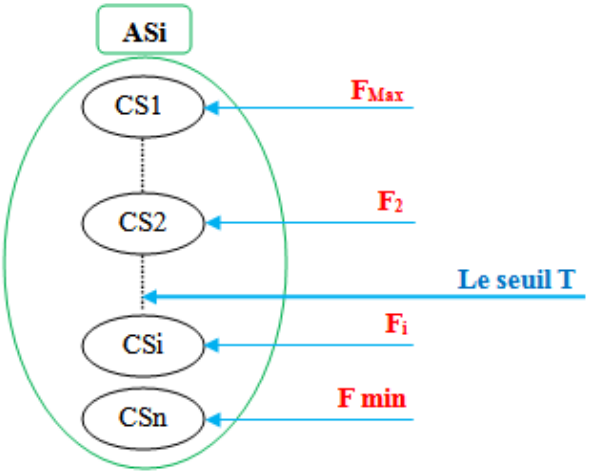

FIGURE 3.5 – Classification des CS dans un AS.

3.4.5.2 La sélection globale

Dans cette section, nous utilisons les services concrets sélectionnés dans la phase de la sélection locale afin de définir toutes les combinaisons possibles pour sélectionner le meilleur service composite en termes de QdS globale, en utilisant une fonction d'utilité " FG " qui sert à calculer la QdS pour chaque service composite tout en respectant les contraintes de QoS globales (GE) imposées par l'utilisateur.

Par la suite, le service composite qui maximise la fonction d'utilité et qui satisfait les GE sera sélectionné.

Tel que :

$$FG = \frac{\sum_{i=1}^{n} Q'_i * W_i}{n} \tag{3.9}$$

Où :

n : représente le nombre d'attributs de QoS du services composite ;

$Q_i^{'}$: représente la valeur normalisée de l'attribut i du service composite ;

W_i : représente le poids exigé par l'utilisateur à l'attribut de QdS j.

La figue 3.6 illustre toutes les étapes de composition par lesquelles passe notre approche.

3.5.6 Surveillance du plan d'exécution

Le contrôle du plan d'exécution est introduit à ce niveau, il permet de garantir une adaptation automatique et une tolérance aux défaillances pouvant survenir durant l'exécution, et ce en remplaçant le service concret qui a échoué par un autre service réalisant la même fonctionnalité. Dans certaines situations, le remplacement devient inopérant en cas d'absence du service concret supposé pallier le problème de défaillance, alors le plan optimal initial sera localement mis à jour à partir du plan global abstrait, cela permet d'éviter la phase de redécouverte de services et la regénération d'un nouveau plan global.

Dans le cas où l'exécution du plan de composition a réussi, le service composite sera publié dans l'annuaire des services concrets pour une utilisation ultérieure.

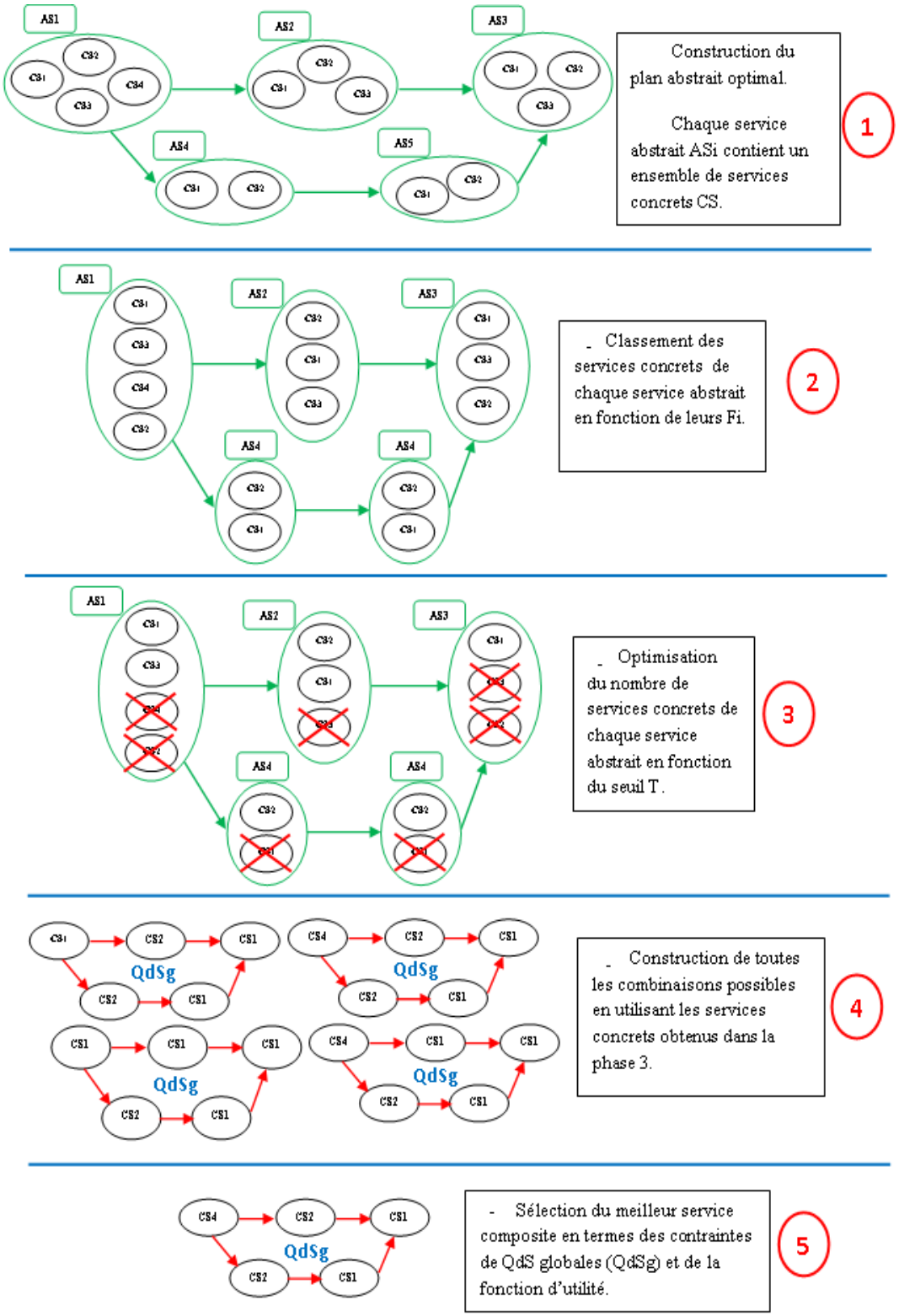

FIGURE 3.6 – Approche de sélection et de composition de services.

3.6 conclusion

Dans ce chapitre, nous avons présenté notre approche de sélection et de composition de services dans les environnements ubiquitaires qui se base sur plusieurs facteurs à savoir la qualité de service locale et globale, les préférences de l'utilisateur, et le contexte d'utilisation.

Afin de valider l'approche proposée, le chapitre suivant sera consacré à la simulation et à l'évaluation des performances à partir des résultats obtebnus.

Chapitre 4

Simulation et evaluation des performances

4.1 Introduction

Dans le chapitre précédent nous avons présenté l'approche (DSSC) proposée pour la sélection et la composition de services dynamique sensible au contexte et aux paramètres de QoS. Dans le présent chapitre, nous présentons les outils utilisés (langage JAVA et XML) ainsi que les résultats obtenus lors de l'implémentation de l'approche proposée. En premier lieu, nous expliquons la démarche adoptée pour valider notre contribution et nous décrivons les étapes principales suivies. Ensuite, nous présentons les résultats de simulation afin d'évaluer ses performances, et cela en la comparant avec l'une des approches proposées dans la littérature.

4.2 Environnement et outils utilisés

L'environnement d'exécution des simulations servant à la validation de la solution proposée, est une machine HP ayant les caractéristiques suivantes :

- Système d'exploitation : Windows 7, 64 bits ;
- Indice de performance Windows : 5.0 ;
- Processeur : Inter(R)Core(MT) i3 CPU M330 @ 2.13GHz 2.13 GHz.
- RAM : 4 Go ;
- Machine virtuelle java 1.6.

Nous avons développé un simulateur écrit en langage JAVA sur la plateforme Eclipse version 1.6 offrant une interface pour simuler une composition de services et exécuter un scénario de sélection des services répondant aux exigences de QoS. De plus, afin de simplifier la représentation des services

abstraits et celle des services concrets, nous avons adopté l'utilisation d'un schéma XML durant la simulation.

4.3 Evaluation des performances

L'évaluation de performances a pour objectif de mesurer le temps de la sélection globale (mesuré en mili-seconds) de l'approche (DSSC) proposée. Il s'agit ici de montrer la stabilité du temps d'exécution, en fixant le nombre de contraintes à sept (07) et en variant le nombre de services abstraits (noté AS) ainsi que le nombre de services concrets (noté CS) dans chaque classe de service abstrait de 10 à 50. Les résultats obtenus sont la moyenne de 20 exécutions, et sont représentés sur la figure suivante :

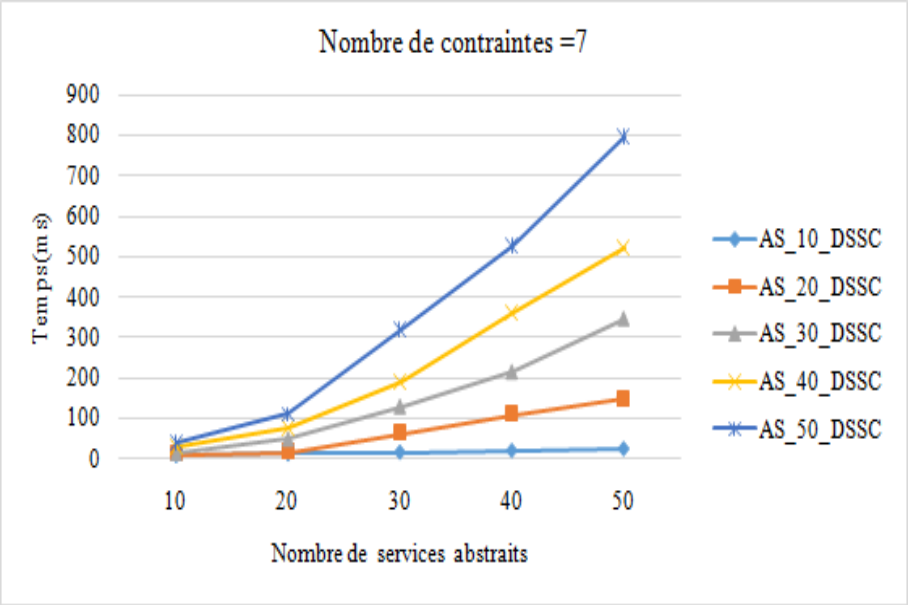

FIGURE 4.1 – temps de sélection vs nombre de service abstrais.

Les résultats de la simulation montrent que notre méthode de sélection globale de services est stable et conduit à un temps de calcul raisonnable. En effet, pour 50 services abstraits et 50 services concrets dans chaque classe de services abstraits, le temps de calcul est inférieur à une (01) seconde. De ce fait, on peut conclure que même en cas de défaillance de l'un des services sélectionné pour la composition, le temps de réexécution du plan sera raisonnable est acceptable.

4.3.1 Comparaison

Pour montrer l'efficacité de notre approche (DSSC), nous avons réalisé une étude comparative avec les résultats obtenus dans [26] . Pour ce faire, nous nous sommes placés dans les mêmes conditions de

simulation en considérant le nombre de services concrets fixé à 50 et le nombre de services abstraits varie entre 10 et 50. Le nombre de contraintes traitées dans [26] étant fixé à cinq (05), notre approche traite aussi le cas de sept (07) contraintes. Les résultats obtenus sont illustrés sur la figure suivante :

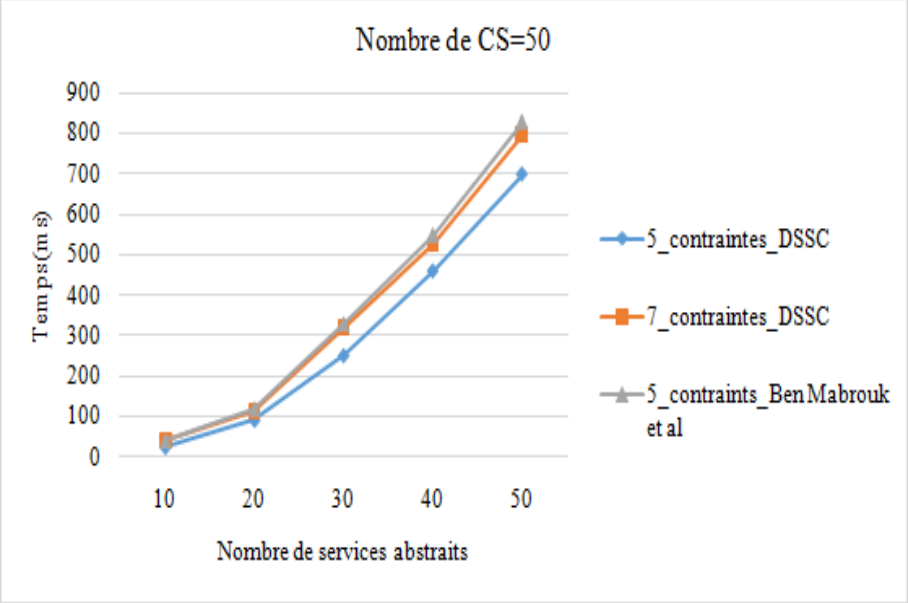

FIGURE 4.2 – Temps de la sélection globale vs. Nombre de contraintes.

La figure montre que le temps de la sélection est influancé par le nombre de contraintes considérées, ceci dit que lorsque le nombre de contraintes augmente le temps d'exécution augmente aussi et ça est due au nombre de traitements efféctués. Les graphes de la figure montrent que l'approche (DSSC) proposée présente de meilleurs résultats, en la comparant à celle de Ben Mabrouk et al, [26], en termes de temps de sélection globale, cela revient à l'utilisation du seuil T qui optimise le nombre de services concraits dans chaque classe de services abstraits. Dans notre contribution nous avons pris en considération deux paramètres du contexte et cinq paramètres de QoS alors que dans [26], seules cinq critères de QoS ont été considérés sans pour autant tenir compte des informations contextuelles.

4.4 Scénario d'application de l'approche proposée

Afin d'illustrer le déroulement de l'approche proposée pour la sélection et la composition de services, nous utilisons un scénario d'assistance et de surveillance à domicile d'une personne dépendante. Il s'agit particulièrement de rappeler les horaires des prises des médicaments pour une personne atteinte de la maladie d'Alzheimer (voir chapitre 1 section 1.1.2).

4.4.1 Description de l'environnement d'exécution du scénario

L'environnement d'exécution du scénario est une maison intelligente composée d'un ensemble de capteurs et d'actionneurs et d'un robot mobile.

- **Caméras de surveillance :** elle se charge de surveiller les activités de la personne dépendante ;
- **Des dispositifs de notification :** Smartphone, PC, haut-parleur (stéréo), TV (écran), etc ;
- **Des capteurs :** ils se chargent de localiser l'endroit ou se trouve la personne ainsi que celui des médicaments ;
- **Robot :** il se déplace pour récupérer les médicaments et de les ramener pour la personne dépendante.

4.4.2 Déroulement du scénario

Les services abstraits utilisés dans notre scénario sont résumés dans le tableau suivant :

Service abstrait (AS)	Entrées	Sorties	Déscription
AS_1	$e_{1,1}$	$s_{1,1}$	Déclencheur horaires
AS_2	$s_{1,1}$	$s_{2,1}$	Localisation du patient
AS_3	$s_{1,1}$	$s_{3,1}$	Localisation du médicament
AS_4	$s_{2,1}$	$s_{4,1}$	Service de notification
AS_5	$s_{3,1}$, $s_{4,1}$	$s_{5,1}$	Déclanchement du robot

TABLE 4.1 – Les services abstraits utilisés dans le scénario illustratif.

Le tableau suivant représente la description des entrées/sorties des services abstraits utilisés.

Entrées	Déscription	Sorties	Déscription
$e_{1,1}$	Vecteur de données : (heure, minute, personne, médicament)	$s_{1,1}$	Signal de localisation
$s_{1,1}$		$s_{2,1}$	Localisation de la personne
$s_{1,1}$		$s_{3,1}$	Localisation du médicament
$s_{3,1}$		$s_{4,1}$	notification qui s'affichera sur : TV, PC, PDA, tablette, etc.
$s_{3,1}$, $s_{4,1}$		$s_{5,1}$	Déplacement du robot

TABLE 4.2 – Déscription des entrées/sorties des services abstraits utilisés dans le scénario illustratif.

Dans le scénario, la requête de l'utilisateur a comme entrées un vecteur de données=<heure, minute, personne, médicament>, et comme sortie le déplacement du robot. R^{in}=$e_{1,1}$ Et R^{out}=$s_{5,1}$.

4.4.3 Plan optimal abstrait de composition

Le plan d'exécution du scénario est obtenu par l'application de l'algorithme de chainage arrière[5] et son optimisation se fait à partir des deux règles : règle d'égalité et la règle d'inclusion simple (voir chapitre 3 section 3.4.2). La figure suivante montre le plan optimal abstrait obtenu :

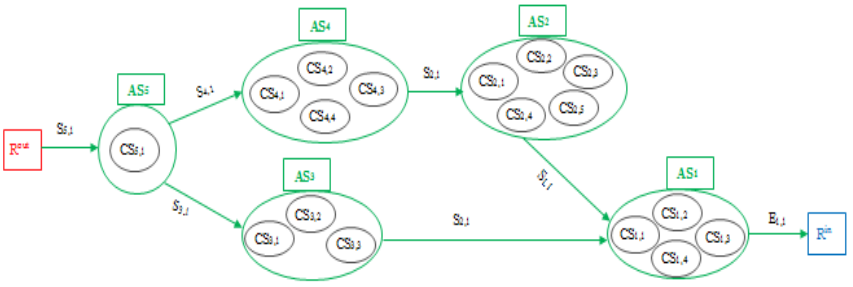

FIGURE 4.3 – Plan optimal abstrait du scénario illustratif.

4.4.4 Sélection locale des services concrets

La sélection locale de service se fait à base du seuil T, on ne garde que les services qui ont une fonction d'utilité $F_i \geq$ T.

Les préférences de l'utilisateur sont 0.2 pour le temps de réponse (TR), 0.2 pour la disponibilité (Disp), 0.2 pour la fiabilité (Fiab), 0.2 pour la sécurité (Sec), 0.2 pour le coût, 0.5 pour l'énergie (Energ) et 0.5 pour la localisation (Loc).

Le tableau suivant montre les valeurs de qualité de service et du contexte des services concrets dans chaque classe de service abstrait.

AS_i	$CS_{i,j}$	TR	Disp	Fiab	Sec	coût	Energ	Loc	QoS	C	F_i	T
AS_1	$cs_{1,1}$	0,9	0,7	0,5	0,8	0,6	0,6	0,8	0,7	0,7	0.7	0.67
	$cs_{1,2}$	0,1	0,2	0,31	0,2	0,26	0,2	0,23	0,21	0,22	0,21	
	$cs_{1,3}$	0,4	0,6	0,2	0,17	0,3	0,23	0,3	0,33	0,26	0,29	
	$cs_{1,4}$	0,54	0,87	0,35	0,34	0,4	0,9	0,84	0,5	0,87	0,68	
AS_2	$cs_{2,1}$	0,1	0,5	0,4	0,4	0,5	0,12	0,9	0,38	0,51	0,44	0,65
	$cs_{2,2}$	0,51	0,2	0,3	0,9	0,35	0	0,18	0,45	0,09	0,27	
	$cs_{2,3}$	0,87	0,35	0,65	0,9	0,7	0,9	0,8	0,6	0,85	0,77	
	$cs_{2,4}$	0,4	0,45	0,6	0,45	0,8	0,1	0,3	0,54	0,2	0,37	
	$cs_{2,5}$	0,65	0,4	0,2	0,7	0,5	0,9	0,8	0,49	0,85	0,67	
AS_3	$cs_{3,1}$	0,2	0,41	0,4	0,6	0,45	0,5	0,23	0,41	0,36	0,38	0,69
	$cs_{3,2}$	0,6	0,1	0,3	0,67	0,65	0.47	0,23	0,46	0,35	0,4	
	$cs_{3,3}$	0,8	0,47	0,76	0,67	0,9	0,8	0,7	0,72	0,75	0,73	
AS_4	$cs_{4,1}$	0,62	0,42	0,36	0,24	0,3	0,3	0,8	0,38	0,55	0,47	0,64
	$cs_{4,2}$	0,32	0,4	0,14	0,23	0,3	0.21	0,84	0,26	0,52	0,39	
	$cs_{4,3}$	0,56	0,2	0,32	0,4	0,21	0,2	0,23	0,34	0,21	0,27	
	$cs_{4,4}$	0,84	0,28	0,3	0,34	0,23	0,3	0,4	0,39	0,35	0,37	
	$cs_{4,5}$	0,85	0,54	0,43	0,84	0,65	0,5	0,9	0,66	0,7	0,68	
AS_5	$cs_{5,1}$	0,32	0,24	0,17	0,54	0,48	0,62	0,2	0,35	0,41	0,38	0,38

TABLE 4.3 – Les valeurs des attributs de QoS pour chaque service concret.

La figure suivante montre le plan obtenu après l'application de l'algorithme de sélection locale :

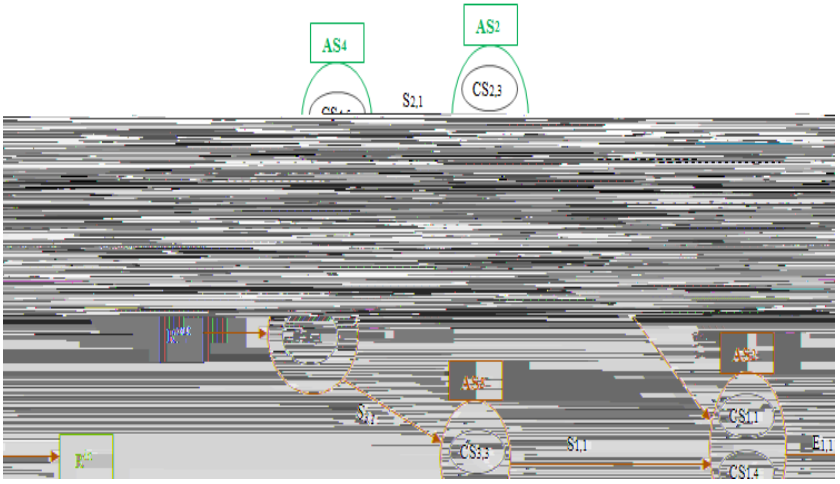

FIGURE 4.4 – Sélection locale des services concrets du scénario illustratif.

4.4.5 Sélection globale

A partir des services concrets issus de la sélection locale, nous allons construire toutes les combinaisons possibles pour sélectionner le meilleur service composite en termes de qualité de service globale. Toutes les combinaisons possibles sont illustré dans la figure suivante :

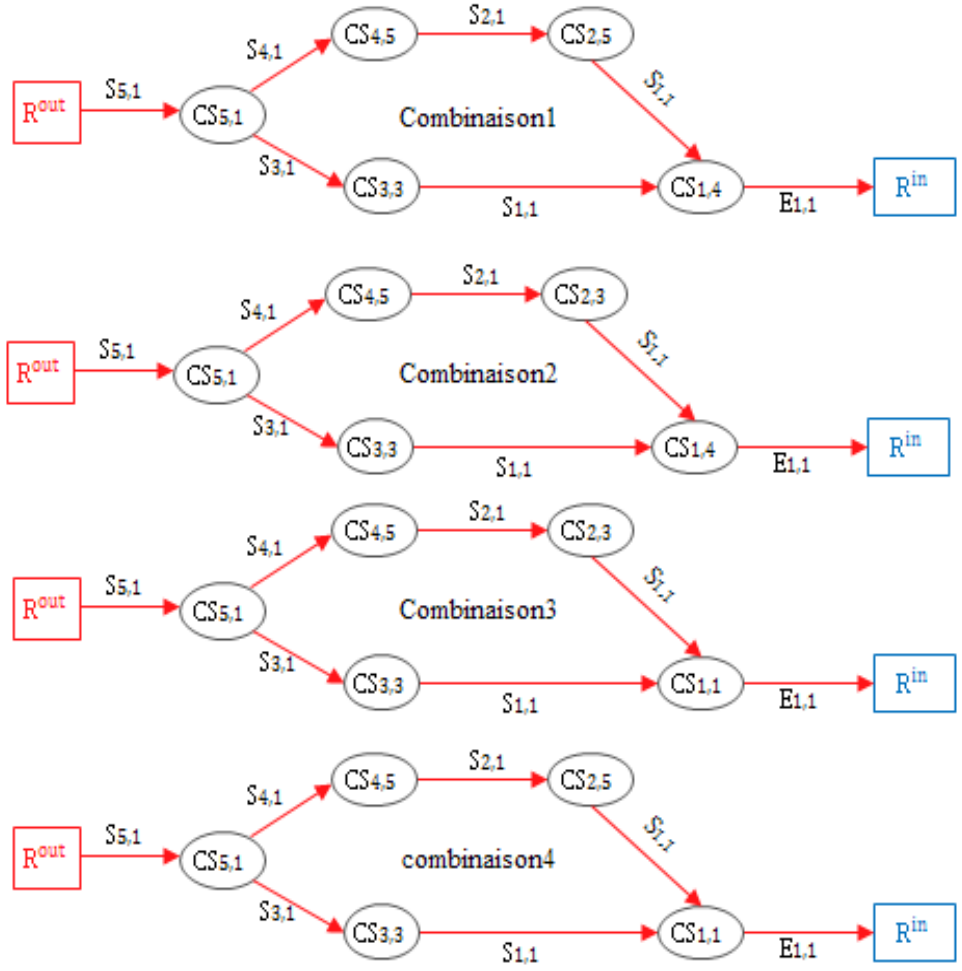

FIGURE 4.5 – Construction de toutes les combinaisons possibles du scénario illustratif.

Le tableau suivant résume les valeurs de la qualité de service globale de chaque combinaison :

Combinaison	Qualité de service globale (FG)
Combinaison1	1,08
Combinaison2	1,21
Combinaison3	1,12
Combinaison4	1,35

TABLE 4.4 – Les valeurs des qualités de service globales pour chaque composition du scénario illustratif.

A partir des résultats du tableau, la meilleure combinaison de services est la 4^{me} avec une qualité de service globale (FG) égale à 1,35.

4.5 Conclusion

Ce chapitre s'est porté sur l'évaluation des performances de notre approche à l'aide d'une série de simulations réalisée en Java. A travers les simulations effectuées, nous avons montré l'efficacité de notre approche et sa supériorité par rapport à l'approche [26].

Conclusion Générale

Les environnements ubiquitaires sont des environnements ouverts et distribués. Ils se caractérisent par des services flexibles et dynamiques (où de nouvelles propriétés peuvent apparaître pour un service comme de nouveaux services peuvent apparaître) et où les besoins des utilisateurs en services varient. Dans un tel contexte, le processus de composition de services pour la satisfaction des besoins de l'utilisateur doit pouvoir s'adapter de manière dynamique aux besoins de ce dernier avec un minimum d'intervention de sa part.

Dans ce travail nous avons proposé une solution pour le problème de sélection et de composition de services dans les environnements ubiquitaires. La principale motivation qui a régit ce travail est la proposition d'une approche dynamique, basée sur la planification de l'intelligence artificiel (IA). Cette approche permet d'une part, d'effectuer une sélection locale en prenant en considération les préférences locales de l'utilisateur en terme de QoS et le contexte d'utilisation, d'autre part, elle vérifie les exigences globales et choisit le meilleur service composite.

Afin de valider notre approche de sélection de services, nous l'avons implémenté sous Java dans le but de mesurer son apport en termes de temps de sélection. Dans tous les tests de simulations effectués, les résultats indiquent que l'algorithme proposé est meilleure que l'approche [26], et ce pour la sélection locale et globale.

Sur la base du travail réalisé, nous dressons les perspectives de recherche suivantes :

- Tester le passage à l'échelle ;
- Implémentation de notre approche dans un environnement réel, en se basant sur une plate forme d'expérimentation .

Bibliographie

[1] : G.Sancho. *Adaptation d'architectures logicielles collaboratives dans les environnements ubiquitaires. Contribution à l'interopérabilité par la sémantique.* Thèse de doctorat en informatique, Université de Toulouse 1, 2010.

[2] : C.Taconet. *Intergiciels pour la sensibilité au contexte en environnement ubiquitaire.* Thèse d'Habilitation à Diriger des Recherches en informatique, Université d'Evry-Val-d'Essonne, France, 2011.

[3] : M.Miraoui. *Architecture logicielle pour l'informatique diffuse : Modélisation du contexte et adaptation dynamique des services.* Thèse de doctorat en Génie Informatique, Ecole de technologie supérieure, Université du Québec, Canada, 2009.

[4] : T.Chaari. *Adaptation d'applications pervasives dans des environnements multi-contextes.* Thèse de doctorat en informatique, Ecole doctorale informatique, Lyon, France, 2007.

[5] : A.Kouicem. *Composition dynamique de services en environnements ubiquitaires.* Mémoire de magister en informatique, Université Abderrahmane Mira de Bejaia, 2008.

[6] : J.Haartsen, M.Naghshineh, J.Inouye , O.Joeressen and W.Allen. Bluetooth : vision, goals, and Architecture. *Mobile Computing and communication review*, 2(4), pages : 38-45, 1998.

[7] : M. Satyanarayanan. Pervasive computing : Visions and challenges . *In proceedings of the international conference on Personal Communications (IEEE)*, pages 10-17, 2001.

[8] : A.Flissi, C.Gransart and P.Merle . Une Infrastructure à Composants pour des Applications Ubiquitaires. *In proceedings of the 2nd French-speaking conference on Mobility and ubiquity computing*, pages : 45-48, 2005.

[9] : H. Pigot, J. Bauchet, and S. Giroux. Assistive Devices for People with Cognitive Impairments. *In The Engineering Handbook of Smart Technology for Aging, Disability, and Independence*, pages : 217-236, 2008.

[10] : L.Gao, X. Zhang and J.Zhu. Preliminary Research on Wearable Healthcare in Ubiquitous Computing Age. *In proceedings of the International Conference on Computer science and software*, vol3, pages : 519-523, 2008.

[11] : M.Doi, k.Ouchi and T.Suzuki. A wearable healthcare support system with timely instruction based on the user's context-advanced motion control : Lifeminder. *In proceedings of the 8th IEEE Internationnal Workshop*, pages : 445-450, 2004.

[12] V.Jones and J.H.Jo. Ubiquitous learning environment : An adaptive teaching system using ubiquitous technology. *In proceedings of the 21st ASCILITE Conference on Beyond the comfort zone*, pages : 474, 2004.

[13] : S.Bellier. *Le E-learning : pédagogie, contenue, modalité, acteurs.* Edition Liaison, page 139, 2001.

[14] : T.Pollet, G.Maas, J.Marien and A.Wambecq. Telecom Services Delivery in a SOA. *In proceedings of the 20th International Conference on Advanced Information Networking and Applications*, pages : 1-5, 2006.

[15] : C.Dumez. *Approche dirigée par les méthodes pour la spécification, la vérification formelle et la mise en oeuvre des services Web composées.* Thèse de doctorat, Université de Technologie de Belfort-Montbéliard, Belfort, France, 2010.

[16] : B.Norman, A.N.Schilit and W.Roy. Context-Aware Computing Applications. *In Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, pages : 85-90, USA, 1994.

[17] : T.Chaari, F.Laforest and A.Flory. Adaptation des applications au contexte en utilisant les services WEB. *In : Proceedings of the 2nd French-speaking conference on Mobility and ubiquity computing*, pages : 111-118, 2005.

[18] : J.Bohn, V.Coroama, M.Langheinrich, F.Mattern and M.Rohs. Living in a world of smart everyday objects. *In Social, Economic, and Ethical Implications*, volume 5, pages : 763-786, 2004.

[19] : K.Madi. *Sélection Globale de Services en Informatique Ubiquitaire.* Mémoire de master recherche, Université de Montpelier II, 2012.

[20] : B.Medjahed, A.Bouguettaya and A.K.Elmagarmid. Composing Web services on the Semantic Web. *The VLDB Journal*, 12(4), pages : 333-351, 2003.

[21] : A.Yachir, *Composition dynamique de services sensibles au contexte dans les systemes intelligents ambiants*, Thèse de doctorat en informatique, l'Université Paris-Est, 2012.

[22] : I.Brakni. *Planification multi-agents pour la composition dynamique.* Mémoire d'ingenieur d'etat en informatique, Université de Tébessa, algerie, 2010.

[23] : D.B.Claro. *Un canevas pour la composition automatique de services web dédiés à la réalisation de devis.* Thèse de doctorat en informatique, Ecole de doctorat d'Angers, France, 2006.

[24] : A.Bekkouche, *Composition des services web sémantiques à base d'Algorithmes Génétiques, Mémoire de Magistère en Informatique*, université de Tlemcen, 2012.

[25] : M. Alrifai and T. Riss. Combining Global Optimization with Local Selection for Efficient QoS-aware Service Composition. *In proceedings of the International conference on Word Wide Web committee (IW3C2)*, pages : 881-980, 2009.

[26] : N. Ben Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas and V. Issarny. QoS-aware Service Composition in Dynamic Service Oriented Environment. *In proceeding of the International Conference on MIDDLEWARE*, pages : 123-142, 2009.

[27] : T.Yu, Y.Zhang and K.J.Lin. Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints. *ACM Transactions on the Web (TWEB)*, 1(1), page 6, 2007.

[28] : L.Zeng, B.Benatallah, A.Ngu, M.Dumas, J.Kalagnanam and H.Chang. QoS-Aware Middleware for Web Services Composition. *Software Engineering, IEEE Transactions*, 30(5), pages : 311-327, 2004.

[29] : F.Moscato, N.Mazzocca, V.Vittorini, G.D.Lorenzo, P.Mosca and M.Magaldi. Workflow Pattern Analysis in Web Services Orchestration : The BPEL4WS Example. *In proceedings of the international conference on High Performance Computing and Communcations*, pages : 395-400, 2005.

[30] : P.Wohed, W.Aalst, M.Dumas and B.Arthur. Analysis of web services composition languages : The case of bpel4ws. *In proceedings of the international conference on Conceptual Modeling*, pages : 200-215, 2003.

[31] : K.Tari, Y.Amirat, A.Chibani and A.Mellouk. Context-aware Dynamic Services Compositon in Ubiquitous Environnement. *In proceedings of the International Conference on communication (ICC)*, pages : 1-6, 2010.

[32] : A.Yachir, K.Tari, Y.Amirat, A.Chibani and A.Badache. MDP and Learning Based Approch for Ubiquitous Services Composition. *In proceedings of the International Conference IEEE*, pages : 1-6. 2010.

[33] : M.Ejaz, H.Mukhtar, D.Belaid and J.B.Song. QoS-Aware Device Selection Using User Preferences for Tasks in Ubiquitous Environnement. *In proceedings of the 7th International Conference on Emerging Technologies (ICET)*, pages : 1-6, 2011.

[34] : A.Yachir, Y. Amirat, A. Chibani and N. Badache. Towards an event-aware approach for ubiquitous computing based on automatic service composition and selection. *Springer Annals of telecommunication Journal*, 67(7-8), pages : 341-353, 2012.

[35] : L.Zhao, Y.Ren, M.Li and K. Sakurai. Flexible service selection with user-specific QoS support in service-oriented architecture. *Journal of Network and Computer Applications*, volume 35, pages : 962-973, 2012.

[36] : S. Chabridon, D. Conan, Z. Abid and C. Taconet. Building ubiquitous QoC-aware applications through model-driven software engineering. *Elsevier Science of Computer Programming Journal*, vol 78, pages : 1912-1929, 2012.

[37] : S.Song and S.Lee. A goal-driven approach for adaptive service composition using planning. *Elsevier Mathematical and Computer Modelling Journal*, vol 58, pages : 261-273, 2013.

[38] : H.Foster, S.Uchitel, J.Magee and J.Kramer. Model-based Verification of Web Service Composition. *In proceeding of the 18th International conference IEEE*, pages : 152-162, 2003.

RÉSUMÉ

Grace à l'apparition du paradigme de l'informatique ubiquitaire, on assiste aujourd'hui à l'émergence de nouveaux systèmes intelligents capables de créer et de gérer des environnements intelligents d'une façon intuitive et transparente. Ces espaces intelligents se caractérisent par l'hétérogénéité et la dynamicité des entités qui les constituent.

Dans ce travail, nous avons proposé une approche qui permet la sélection et la composition dynamique de services dans les environnements ubiquitaires. Cette approche prend en compte, d'une part, le contexte d'utilisation et la QoS offerts par les services disponibles et d'autre part, les préférences des utilisateurs. Pour évaluer les performances de la solution proposée, nous avons réalisé des simulations en Java. Les résultats obtenus ont été comparés avec l'une des approches proposées dans la littérature, ils montrent une amélioration en termes de temps de séléction.

Mots-clés :

Informatique ubiquitaire, Composition et sélection de services, QoS, Contexte, SOA.

ABSTRACT

By dint of the emergence of ubiquitous computing paradigm, we are witnessing the emergence of new intelligent systems able to create and manage intelligent environments by an intuitive and transparent way. These smart spaces are characterized by heterogeneity and dynamic entities that constitute them.

In this piper, we proposed an approach which allows the dynamic services selection and composition taking into account, firstly, the context of use and the QoS offered by the services available in a ubiquitous environment and secondly, user preferences. To evaluate the performance of this solution, we performed simulations in Java. The results obtained were compared with one of the approaches proposed in the literature ; they show an improvement in terms of selection time.

Keywords :

Ubiquitous Computing, Service composition and selection , QoS, Context, SOA.